... books look like books because, though it sounds a bit simple to say so, that's what they are. And that's what they're supposed to look like.

Richard Hendel
"A book designer's odyssey"
Scholarly Publishing,
July 1986, p. 350

TUGBOAT

THE TEX USERS GROUP NEWSLETTER
EDITOR BARBARA BEETON

VOLUME 7, NUMBER 3 • OCTOBER, 1986
PROVIDENCE • RHODE ISLAND • U.S.A.

Addresses

James Alexander
Department of Mathematics
University of Maryland
College Park, MD 20742
Arpanet: alex@eneevax.umd.edu

Jacques André
IRISA University of Rennes
Campus de Beaulieu
F-35042 Rennes Cedex, France
99-364815

Richard L. Aurbach
Monsanto Company
800 N. Lindbergh Blvd.
St. Louis, MO 63167
314-694-5453

Lawrence A. Beck
Grumman Data Systems
R & D, MS D12-237
Woodbury, NY 11797
516-682-8478

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
Arpanet: bb@Sail.Stanford.Edu,
bnb@xx.lcs.MIT.Edu

Charles Bigelow
Bigelow & Holmes
15 Vandewater Street
San Francisco, CA 94133
415-788-8973

Malcolm Brown
ACIS/IRIS
Stanford University
Cypress Hall, Rm E7
Stanford, CA 94305
415-723-1055
Arpanet: MBB@SU-Lindy

Lance Carnes
163 Linden Lane
Mill Valley, CA 94941
415-388-8853

S. Bart Childs
Dept of Computer Science
Texas A & M University
College Station, TX 77843-3112
409-845-5470

Maria Code
Data Processing Services
1371 Sydney Dr
Sunnyvale, CA 94087
408-735-8006

John M. Crawford
Computing Services Center
College of Administrative Science
Ohio State University
Columbus, OH 43210
614-422-1741
CSNet: Crawford-J@Ohio-State
Bitnet: TSO135@OHSTVMA

Jackie Damrau
Dept of Math & Statistics
Univ of New Mexico
Albuquerque, NM 87131
505-277-4623
Bitnet: damrau@unmb
UUCP: damrau@unmvax

Hans Ehrbar
Economics Department
University of Utah
Salt Lake City, UT 84112

Maureen Eppstein
Administrative Publication Manager
Stanford University
Encina Hall, Room 200
Stanford, CA 94305
415-723-9254
Arpanet: MVEppstein@Score.Stanford.Edu

Michael J. Ferguson
INRS - Télécommunications
Université du Québec
3 Place du Commerce
Verdun (H3E 1H6), Québec Canada
514-765-7834

Jim Fox
Academic Computing Center HG-45
University of Washington
3737 Brooklyn Ave NE
Seattle, WA 98105
206-543-4320
Bitnet: fox7632@uwacdc

David Fuchs
Department of Computer Science
Stanford University
Stanford, CA 94305
415-723-1646
Arpanet: DRF@Score.Stanford.Edu

Richard Furuta
Department of Computer Science
Univ of Maryland
College Park, MD 20742
301-454-1461
Arpanet: Furuta@Maryland

Raymond E. Goucher
TEX Users Group
P. O. Box 9506
Providence, RI 02940
401-272-9500 x232

William Gropp
Dept of Computer Science
Yale University
Box 2158 Yale Station
New Haven, CT 06520
203-436-3761
Arpanet: Gropp@Yale

Dean Guenther
Computer Service Center
Washington State University
Computer Science Building,
Room 2144
Pullman, WA 99164
509-335-0411
BITnet: Guenther@WSUVM1

Klaus Guntermann
Technische Hochschule Darmstadt
Fachbereich Informatik
Inst für Theoretische Informatik
Alexanderstraße 24
D-6100 Darmstadt
Federal Republic Germany
Bitnet: gunterma at ddadvs1

Doug Henderson
Division of Library Automation
Univ of California, Berkeley
186 University Hall
Berkeley, CA 94720
Bitnet: dlatex@ucbcmsa

Amy Hendrickson
57 Longwood Ave #8
Brookline, MA 02146
617-738-8029

Alan Hoenig
574 Argyle Rd
Brooklyn, NY 11230
718-856-3696

Thomas Hofmann
Ciba-Geigy AG
R-1032.5.68
P. O. Box
CH-4002 Basle, Switzerland
+41 61/37 30 59
uucp: ...!mcvax!cernvax!mdtch!sys2!tom

Patrick D. Ion
Mathematical Reviews
416 Fourth Street
P. O. Box 8604
Ann Arbor, MI 48107
313-996-5273

Rick Jansen
Academic Computing Services
Amsterdam (SARA)
P. O. Box 4613
1009 AP Amsterdam, Holland
+31-20-5920242; +31-20-5923000
Bitnet: Rick@HASARA5

Helmut Jürgensen
Dept of Computer Science
Univ of Western Ontario
London N6A 5B7, Ontario, Canada
519-679-3039
Bitnet: A505@UWOCC1
UUCP: helmut@deepthot

Arthur Keller
University of Texas at Austin
Department of Computer Science
Austin, TX 78712-1188
512-471-7316
Arpanet: ARK@SALLY.UTexas.Edu

Donald E. Knuth
Department of Computer Science
Stanford University
Stanford, CA 94305
Arpanet: DEK@Sail.Stanford.Edu

Leslie Lamport
Systems Research Center
Digital Equipment Corp
130 Lytton Ave
Palo Alto, CA 94301
415-853-2170
Arpanet: lamport@SRC.DEC.COM

John Lee
jslee@nrtc.arpa

Pierre A. MacKay
Northwest Computer Support Group
University of Washington
Mail Stop DW-10
Seattle, WA 98195
206-543-6259; 545-2386
Arpanet: MacKay@Washington

Rick Mallett
Computing Services
Room 1208 Arts Tower
Carleton University
Ottawa (K1S 5B6), Ontario, Canada
613-231-7145

Laurie Mann
Stratus Computer
55 Fairbanks Boulevard
Marlboro, MA 01752
617-562-4061

Yoshio Ohno
Institute of Information Science
Keio University
1–1 Hiyoshi 4-chome
Kohoku-ku Yokohama
223 Japan

Richard S. Palais
Department of Mathematics
Brandeis University
Waltham, MA 02154
617-647-2667

Mitch Pfeffer
Suite 90
148 Harbor View South
Lawrence, NY 11559
516-239-4110

Arnold Pizer
Department of Mathematics
University of Rochester
Rochester, NY 14627
716-275-4428

Craig Platt
Dept of Math & Astronomy
Machray Hall
Univ of Manitoba
Winnipeg R3T 2N2
Manitoba, Canada
204-474-9832
platt%cc.uofm.cdn@ubc.csnet

Vincent Quint
Laboratoire de génie informatique
IMAG, BP 68
F-38402 St. Martin d'Hères Cedex,
France
mcvax!imag!quint@seismo

Tom Rokicki
Computer Science Dept
Stanford University
Stanford, CA 94305
Arpanet: Rokicki@SU-Sushi

Wolfgang Rülling
Universität des Saarlandes
Fachbereich Angewandte Math und
Informatik, SFB 124
Postfach
D-6600 Saarbrücken
Federal Republic Germany

John Sauter
801128 Bates Road
Merrimack, NH 03054
603-881-2301

Joachim Schrod
Technische Hochschule Darmstadt
Fachbereich Informatik
Inst für Theoretische Informatik
Alexanderstraße 24
D-6100 Darmstadt
Federal Republic Germany

E. W. Sewell
3822 Hillsdale Lane
Garland, TX 75042-5334
214-272-0515

Barry Smith
Kellerman & Smith
534 SW Third Ave
Portland, OR 97204
503-222-4234

Alan Spragens
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014

Ralph Stromquist
MACC
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706
608-262-8821

Rilla Thedford
Intergraph Corporation
MS HQ013
One Madison Industrial Park
Huntsville, AL 35807
205-772-6494

Klaus Thull
c/o Dittrich
Zeughofstraße 23
D-1 Berlin 36
Germany

Georgia K.M. Tobin
The Metafoundry
OCLC Inc., MC 485
6565 Frantz Road
Dublin, OH 43017
614-764-6087

Joey K. Tuttle
I P Sharp Associates
220 California Avenue
Suite 201
Palo Alto, CA 94306
415-327-1700

Samuel B. Whidden
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500

Reinhard Wonneberger
Drachenstieg 5
D 2000 Hamburg 63
Federal Republic Germany
040/592354
Bitnet: BO3WBG at DHHDESY3

Ken Yap
Dept of Computer Science
University of Rochester
Rochester, NY 14627
Arpanet: Ken@Rochester
Usenet: ..!{allegra,decvax,seismo,
cmcl2,harvard,topaz}
!rochester!ken

Hermann Zapf
Seitersweg 35
D-6100 Darmstadt
Federal Republic Germany

General Delivery

From the President

Bart Childs

We have just concluded the '86 meeting at Tufts. It was a major step to hold the meeting at a location other than our Mecca. The differences were more than just a few thousand miles. Commercial eateries, watering holes, and hotels were not close. Still, the meeting was GREAT! Bill Schlessinger was a gracious and hospitable host. The dormitory arrangements received many compliments within my earshot. I never ate with the *real* people at the meeting because we always had a *Steering Committee* meeting during the lunch break, in a more private surrounding. I heard many good things about those meals and ours were good too.

Rilla Thedford did another excellent job of pulling a program out of our hesitant membership. Ray Goucher was *Johnny on the spot* and Karen Butler did her usual excellent job of keeping Ray and the steering committee pointed in the right direction. We are fortunate to have these dedicated employees and the great assistance of Sam, Barbara, and others at AMS. We owe them our gratitude and a lot more.

The meeting had more PC flavor than ever before, as one should expect. Our attendance was down a little. We desperately need a few good people to volunteer. Typical jobs for volunteers include:

- Annual meeting organizer, as in call for papers, etc. See a later item.
- Local host for annual meeting.
- Accounting and budgeting helper.
- Legal services; we need to become incorporated.
- Corporate relations, donations, fund raising.
- Volunteers to coordinate and help promote TUG courses. Several courses have been cancelled due to insufficient enrollment. We feel that just a little contact and selling would make these healthy. The income helps us all. We probably need an overall coordinator and then regional ones as well.
- Volunteer to solicit, coordinate, and encourage other volunteers.
- Public relations and advertising volunteer.
- Volunteers to hassle president, site coordinators, etc., for getting input to TUGboat, etc.
- A local chapter coordinator and a book on how to form one, etc.

A lot of people have volunteered to work, in response to Pat Fina's questionnaire. We will begin extracting names from that list soon. If one of the above items strikes your fancy, send Ray your name, best way to contact you, and your preferred assignment. It is best to do both a phone call and a written follow-up. I would appreciate a copy.

Here are some items from the Steering Committee meetings:

- The next meeting will be planned for the West Coast, perhaps at Stanford. The sentiment was that we had a great meeting and should move around some. This issue of TUGboat has a call for papers and a suggested session topic. Pierre MacKay, Joey Tuttle, and Richard Palais are the program committee. We feel the first day would include status reports, help sessions, vendor displays, and other "helpful" information. It is hoped that two more days with reviewed papers might be appropriate. Since TEX and METAFONT are stable, our society will likely become more like AMS, etc. **Remember that one of our biggest problems is individual timidity in the submission of papers and ideas to the meetings and TUGboat!**
- We are in better financial shape than last year. This has been due to good efforts by Ray, Barbara, Karen, and Stephan v. Bechtolsheim.
- Regional courses were taught at Harvard, Vanderbilt, Illinois Institute, and Stanford. We had to cancel 5 sets of courses due to lack of enrollment. *Just a few volunteers drumming up regional business could have made this more successful.* Twelve weeks of in-house courses have been taught at research laboratories, universities, and private companies.
- Our membership trends are shown by the following table:

	1983	1984	1985	1986
Members/Subscribers	755	1123	1498	2125
Institutional members	44	76	96	113
Annual meeting attend.	135	153	139	145
TUG courses taught	1	2	18	31

(Courses varied in length from two or three to five days.)

- Rilla Thedford and Allan Hoenig were elected Vice-President and Secretary, respectively. Thanks to Nelson Beebe, John Gourlay, and Richard Palais for preparing a slate of candidates. Rilla's election created an at-large vacancy in the Finance Committee. Amy Hendrickson was selected and has accepted that role.

- Ray Goucher will increase his contact with site coordinators to be more aggressive in pursuing potential TUGgers.
- We wish to acknowledge the following contributions to our society: Donald Knuth $6,441 from his royalties for *The TEXbook* and *The METAFONTbook*; Kellerman and Smith for contributing $905.16 for the first Donald Knuth scholarship; and Textset $1,000.
- Approval was given to hire a technical assistant for the TUG office. See the last page of this issue for a description of the qualifications for this position.

Donald E. Knuth Scholarship

The first Donald E. Knuth Scholarship was awarded this year to Jackie Damrau, an Editorial Assistant in the Department of Mathematics and Statistics at the University of New Mexico. The award included all expenses associated with attendance at the annual TUG meeting at Tufts, and at the two-day course on output routines which followed the meeting. Jackie is especially interested in uses of and extensions to LaTeX, and at the meeting volunteered to edit a TUGboat column on LaTeX. She has communicated to us the following news in connection with the scholarship award:

> "On a personal note, the Donald E. Knuth Scholarship enabled me to obtain a promotion. When I came to work on Monday, July 28, I was informed that my promotion was put through because of the accomplishment of proving that my work is invaluable to the department. I thank everyone for the honor of winning the scholarship."

Congratulations to Jackie on her accomplishments.

The Scholarship was funded by a grant from Kellerman & Smith of Portland, Oregon. TUG's thanks go to K&S for their generosity, and also to this year's Scholarship Committee: Maria Perkins of Vanderbilt University (chair), Carl Smith of the University of Maryland, and Linda Woessner of the NASA Langley Research Center.

One or more Knuth Scholarships will be awarded next year. Although next year's committee has not yet been named, nor have deadline dates been set, it is not too early for prospective applicants to begin designing their submissions.

The intent of the Knuth Scholarship is to encourage the increase of knowledge about TeX and to sharpen the TeX skills of non-technical users. The competition will be open to all 1987 TUG members holding support positions that are secretarial, clerical or editorial in nature. As this year, the award will consist of an all-expense-paid trip to TUG's 1987 Annual Meeting and the two-day course offered in conjunction with the meeting.

Submissions should include two applications of TeX: a TeX'ed resume, and the input and final TeX output of a project that displays originality, knowledge of TeX, and good TeXnique. The project may make use of a macro package, either a public one such as LaTeX or one that has been developed locally; such a macro package should be identified clearly. Such features as sophisticated use of math mode, of macros that require more than "filling in the blanks", or creation and use of new macros will be taken as illustrations of the applicant's knowledge.

Further requirements, dates, and the subject of next year's short course will be published in the next issue of TUGboat.

Barbara Beeton

1987 Annual Meeting, Call for Papers: TEX for the Humanities

Pierre MacKay
University of Washington

Editor's note: The Steering Committee at the last TUG meeting charged the Program Committee for the 1987 Annual Meeting to select and arrange a core program with a coherent theme; the papers presented within this framework are to be refereed and published in a volume of proceedings. The topic chosen, TEX for the Humanities, is expected to attract a different spectrum of users than have attended TUG meetings in the past; the invitation below will be extended to members of various organizations in the humanities as well as to present TUG members. In addition to the thematic program, the overall program will continue to include sessions on issues of implementation, hardware and software support, question and answer sessions, and other topics of general and technical interest.

Now that Donald Knuth's TEX typesetting system has become available on personal desktop computers such as the IBM PC and the Macintosh, the TEX user comunity is fast growing beyond its original nucleus of technically oriented users. Humanists in particular are coming to understand how TEX will provide a means of quality publication free from the constraints that have limited our use of foreign language texts and non-Latin character sets. The METAFONT program (part of the overall TEX system) is already available on a wide variety of computer systems and will soon reach the same personal desktop machines as now run the TEX program. Using METAFONT, we shall be able to supply text fonts in languages which have long been excluded from North American scholarly publication, and have become rarer even in European scholarly publication.

In recognition of this development, the TEX Users Group (TUG) is issuing a call for papers to be presented at the TUG general meeting on the West Coast, on a date during the last half of July or the first week of August. Depending on the response to this call, we shall organize an afternoon, a full day, or a full program of papers with the general theme *TEX for the Humanities*.

We urge that you give this meeting your special consideration. The TUG meetings are attended by a large group of commercial vendors who offer typesetting and proof-copy machines, font support, programming support, and book quality typesetting services. The best way to persuade these vendors to take an interest in the needs of the Humanities is to attend this meeting and make our presence felt.

Contributions will be judged for selection by the Program Committee and should show evidence of a good basic knowledge of the workings of TEX or METAFONT or both. The committee is equally interested in projects which have already achieved some measure of success and in designs for new projects. Papers selected by the committee will be eligible for publication in the Proceedings of this meeting, in a format to be announced at a later date.

The following deadlines will apply:

Title and short abstract	October 31, 1986
First selection	November 30, 1986
Long abstract	January 31, 1987
Firm selection	February 28, 1987
TEX copy for preprint	April 15, 1987
Final copy in machine readable form (not camera ready)	Last day of TUG meeting

Contributions should be submitted to a member of the Program Committee:

Pierre MacKay, University of Washington
Richard Palais, Brandeis University
Joey Tuttle, I. P. Sharp Associates

Report from ANSI X3V1

Lawrence A. Beck
Grumman Data Systems

ANSI X3V1 has been very busy since the last TUG meeting. As a result of our labors, the *Standard Generalized Markup Language* (SGML) has become an official International Standard (its official designation is ISO 8879) and the national ANSI adoption procedures are well underway. In addition, SGML has been provisionally adopted by the U.S. Department of Defense as the markup standard for all text used in interchange. The first instance of DoD citing this standard is in the *Proposed Military Standard for Automated Interchange of Technical Information* issued by the U.S. Air Force Logistics Command in conjunction with the CALS (Computer Aided Logistics System) Project.

DoD adoption of SGML means that the use of SGML will be a requirement in any DoD procurement (and eventually in all U.S. Government procurements) that involves interchange of text.

Since SGML is not a formatting language, this doesn't pose a problem for TUG. In fact, there are efforts already in progress to build SGML front end parsing systems that interface SGML to TEX. The first parser of this type has been built by Mr. Craig Smith of the Hahn-Meitner Institute (DM) in West Berlin. Dr. Alan Spragens of SLAC is also considering a project along the same lines; Alan has been invited to attend ANSI and ISO meetings and he may do so.

Copies of the SGML Standard can be obtained from

American National Standards Institute
1430 Broadway
New York City, NY 10018

Request ISO 8879 SGML. I believe that ANSI charges for these copies as it is a copyrighted document (ISO holds the copyright).

In addition, there are other projects underway which might be of interest to TUG, specifically, a project on Text Composition. This is a two-part project, the first covering *Text Composition Semantics and Syntax* and the second part being a standard for a *Text Presentation Metafile*.

The text presentation metafile may be of interest and/or concern to TUG as this is akin to the TEX `DVI` file. Two of the main participants in this effort are Adobe (who market PostScript) and Xerox (who are attempting to market Interpress). I believe that the subgroup would welcome input from TUG. As Grumman does not have an interest in the development of this standard, I don't keep a hand in the work of this group. However, if TUG has a position that relates to it, I will be happy to relay it.

If anyone would like further information on these, or on other ANSI and ISO projects, they can contact me at my office. My phone number is 516-682-8478.

Towards a TEX Philology Group

Reinhard Wonneberger

These last times, there has been an increasing number of communications concerning the use of foreign and even exotic languages under TEX and its derivatives. While some of these efforts are backed up by larger institutions, others are more or less private. So there seems to be a growing amount of interest in exchanging information and cooperation in this area.

From a TEX point of view, *levels of effort* in this area quite naturally fall into the following categories:

- just using what is available
- design of foreign character sets
- macro support using TEX as it is
- modifications to TEX

From a *problem* point of view, the following topics seem to be of special interest:

- accenting
- hyphenation and alternatives like characters differing in width
- producing right-to-left text like a phone book in Arabic
- quoting right-to-left text like Hebrew in a Latin environment
- handling symbol scripts like Chinese

These languages, among others, are involved:

- Arabic
- Devenagari
- Greek
- Hebrew
- Russian
- Polish

To get started, we had an informal session at the Strasbourg TEX conference of June 19–21, which was mostly concerned with the question of how such a group could be organized. We compiled an ad-hoc

address list of people who were present and made it available to them.

To come to a sounder basis that might eventually become a TEX philology group, we suggest that the following information be provided by interested individuals and institutions:

1. Information for a more detailed address list:
   Name and address
   Affiliation
   Phone number
   Network address
   Level of effort
   Problems
   Languages
2. A short description of any project you have done, are doing, will do, or have heard of in any of these fields.
3. Bibliographic references, preferably with short comments.

It will also be interesting to know something of your concerns as an author, publisher, typographer, informatician, etc.

The material you supply, including address information, may be either published directly, or handed over to some volunteer to write an overview article.

Responses may be sent by either regular or electronic mail, to

Reinhard Wonneberger
Drachenstieg 5
D 2000 Hamburg 63
Federal Republic Germany
Bitnet: `BO3WBG at DHHDESY3`

or to

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940, U.S.A
Arpanet: `BNB@XX.LCS.MIT.EDU`

An Idea Exchange on TEX Training

Laurie Mann
Stratus Computer

At the July TUG meeting, a number of us discussed issues related to the training of new TEX users. As a vehicle for exchanging ideas on TEX training and TEX macros, I volunteered to establish a publication called TEXtrain. The first issue was due out in mid-September, but since I had only two contributions (one of which was mine), I decided to hold off for a few months, to see if any other TUG members would be interested in contributing.

TEXtrain was originally conceived of as an "amateur press association" (APA for short). Contributions are sent to a central person, who then distributes them to other members of the APA. To remain a member of the APA, each member must contribute something to the APA every few issues. APAs are generally self-supporting — members pay for their own postage and repro costs.

Another way to handle the dissemination of training information might be to start a column on TEX training in a future issue of TUGboat. Barbara Beeton and I have discussed this possibility, and will probably continue to do so in the future.

If you have any feelings as to how this should be handled (by starting a specialized publication, by running a regular column in TUGboat, or by some other means), please write to me,

Laurie Mann
c/o Stratus Computer
55 Fairbanks Boulevard
Marlboro, MA 01752

or contact Barbara Beeton.

Software

Another Approach to Multiple Changefiles

Klaus Guntermann and Wolfgang Rülling*
Technische Hochschule Darmstadt

As reported by W. Appelt and K. Horn in TUGboat Vol. 7, No. 1, pp. 20–21, there are several reasons to allow multiple changefiles in the development of a WEB program. These need not to be repeated here. But we did not follow the same approach when we faced the problem. We decided to develop a separate program which we called TIE, since it ties several parts of a WEB together.

Furthermore we allow that a changefile modifies parts that were just changed. The general strategy is that the addition of changefile f_{i+1} behaves as if the changefiles f_1 to f_i had been merged into the WEB program before.

We use a separate program because of the following reasons:

- This single simple program makes additional changes to two programs (namely TANGLE and WEAVE) unnecessary.
- A WEB software developer needs a tool to incorporate frozen changes into a new release of his WEB program from time to time. If the preprocessor program can either create a single changefile or merge all changes into a new WEB file modifications can be written and tested via an additional changefile without touching the released source file. Finally the changes are added ("tied") to the new release.
- TIE can be used for other WEB like systems, too, e.g. for a C version of WEB we created recently. Furthermore TIE allows the application of the changefile method to "plain" Pascal (or even FORTRAN, or whatever programming language you like). One can just merge the changes into the program by selection of the "create new WEB file" option. This is possible since TIE just knows about the line oriented structure of changefiles and has not to deal with the WEB control sequences for sections and so on in the main WEB file. Even data files — as long as they contain textual data — might be changed this way.

The only drawback compared to the method suggested by Appelt and Horn seems to be that it introduces another preprocessing step. This takes some time when the changefiles for large programs like TEX or METAFONT have to be tied since the complete WEB source must be read once more.

Comments to our approach are welcome. TIE is available as a WEB program and can be obtained for a handling charge from Klaus Guntermann at Technische Hochschule Darmstadt.

* now at Universität des Saarlandes, Saarbrücken

WEB Adapted to C

Klaus Guntermann and Joachim Schrod
Technische Hochschule Darmstadt

In the UNIX environment the programming language C usually is best developed. For systems programming it seems even to be more suitable than Pascal. This led us to the development of CWEB that allows literate programming in C, giving the full documentation tools that WEB adds to Pascal.

The CWEB processors (simply named by a C- prefix, as is the whole system — there was no common relative whose initials the implementors could choose) CTANGLE and CWEAVE are derived from their WEB counterparts.

Changing TANGLE to build a C program instead of a program for the Pascal compiler was rather straightforward. The only problems occurred with the C preprocessor statements that must be allowed in a CWEB program. These statements are supposed to start on a new line, may span several lines ending with a backslash, and in the last line (which may be the first) no other text is allowed to follow. With new tokens designating start and end of a preprocessor statement it was rather easy to add the necessary rules.

Adapting WEAVE to parse C was a heavier task. One of the reasons is that the beginning of a C function declaration cannot be detected very easily since declaration and call of a function look similar if one does not look ahead very far. The look ahead is nearly impossible if CWEB sections are used for the parameter declaration or the function body. We introduced a new (i.e. in addition to WEB) control sequence @h that marks the start of a function heading in a declaration.

The grammar had to be rewritten completely. We tried to overcome some of the problems that WEAVE has with Pascal formatting if there is not a bunch of explicit formatting commands. The

goal was to get satisfactory results even if explicit formatting is seldom used. (So we are not happy that we had to introduce the control sequence for function headings!)

The CWEBMAC file that specified the layout of the formatted output was redesigned. One major aim was to allow modification of the TEX layout parameters, e.g. tolerance, penalty and paragraph formatting within the documentation parts of the program. (Modification of the tolerance parameter is essential if you try to comment a program in German unless you accept a lot of over-/underfull hbox messages.) In addition one can choose the amount of indentation in the program part.

Following we give a CWEB version of the probably well known UNIX wc command that counts lines, words and characters in a (list of) file(s).

CWEB is currently written in WEB. A rewrite in CWEB is planned but not yet started.

CWEB can be distributed for a handling charge of DM 150.00 (US-$ 60.00) for educational/research sites, DM 250.00 (US-$ 100.00) for others, on magnetic tape (1600 or 6250 bpi, EBCDIC or ASCII, please state format) or DOS 360 K byte floppy disk.

1. CWEB-**Example.** This example presents the "word count" program from UNIX. We rewrote it in CWEB to demonstrate literate programming in C.

2. Most CWEB programs share a common structure. Differences are found merely when all functions are just introduced when needed without any special sequence.

```
⟨ global #includes 3 ⟩
⟨ global variables 4 ⟩
⟨ all functions 17 ⟩
⟨ main 6 ⟩
```

3. We must include the standard I/O definitions to use formatted output to *stdout* and *stderr*.

```
⟨ global #includes 3 ⟩ ≡
#include ⟨stdio.h⟩
```

This code is used in section 2.

4. As global variables we introduce some counters that take the cumulated values for each file.

```
⟨ global variables 4 ⟩ ≡
    long wordct, linect, charct;
```

See also section 5.

This code is used in section 2.

5. In case that we have to process a list of files we must sum up the grand total in another set of variables.

```
⟨ global variables 4 ⟩ +≡
    long twordct, tlinect, tcharct;
```

6. Now we come to the general layout for the *main* function.

```
⟨ main 6 ⟩ ≡
main (argc, argv)
    int argc;
    char **argv;
{
    ⟨ local variables of main 16 ⟩
    ⟨ set up option selection 7 ⟩
    ⟨ go and process all the files 8 ⟩
    ⟨ add the grand total line if there were multiple
            files 15 ⟩
    exit(status);
}
```

This code is used in section 2.

7. The first argument should be able to select the counters the user needs. Each selection is given by the initial character (lines, words or characters). We do not process this option string now. It is sufficient just to suppress unwanted figures at output time. However, if no such option was given, print all three values.

```
⟨ set up option selection 7 ⟩ ≡
    wd = "lwc";
    if (argc > 1 ∧ *argv[1] ≡ '-') {
        wd = ++argv[1]; argc--; argv++;
        }
```

This code is used in section 6.

8. Now we scan all the arguments and try to open a file, if there is an entry left. The file is processed and its statistics are written. We update the grand total counts anyway.

```
⟨ go and process all the files 8 ⟩ ≡
    i = 1;
    do {
        ⟨ if a file is given we should try to open
                it 9 ⟩
        ⟨ initialize pointers and counters 10 ⟩
        ⟨ scan this file 11 ⟩
        ⟨ write this file's statistics 13 ⟩
        close(f); ⟨ update grand totals 14 ⟩
        }
    while (++i < argc);
```

This code is used in section 6.

9. Now we should open the next file. A special trick allows us to handle input from *stdin*—usually file number 0—if no file name was given at all. In this case we use the preset value 0 for f to read input from. If we could not open a file, we set the return code for the program an try to proceed for the other parameters.

```
⟨ if a file is given we should try to open it 9 ⟩ ≡
    if (argc > 1 ∧ (f = open(argv[i], 0)) < 0) {
        fprintf(stderr, "wc: cannot open %s\n",
                argv[i]); status = 2; continue;
        }
```

This code is used in section 8.

10. Since buffered I/O speeds up things very much we use it, but we do the buffering ourselves. This means that we have to set up pointers and counters such that something is read into the buffer on the first try.

```
⟨ initialize pointers and counters 10 ⟩ ≡
    p1 = p2 = b; linect = 0; wordct = 0;
    charct = 0; token = 0;
```

This code is used in section 8.

11. Now we scan the file. The *token* variable indicates if we are just within a word.

```
⟨ scan this file 11 ⟩ ≡
    while (1) {
        ⟨ fill buffer if it is empty 12 ⟩
        c = *p1++;
        if (' ' < c ∧ c < '177') {
            if (¬token) {
                wordct++; token++;
                }
            continue;
            }
        if (c ≡ '\n') linect++;
        else if (c ≢ ' ' ∧ c ≢ '\t') continue;
        token = 0;
        }
```

This code is used in section 8.

12. Using buffered I/O makes it very easy to count the number of characters in the file, almost for free.

```
    define BUFFERSIZE = 512
⟨ fill buffer if it is empty 12 ⟩ ≡
    if (p1 ≥ p2) {
        p1 = b; c = read(f, p1, BUFFERSIZE);
        if (c ≤ 0) break;
        charct += c; p2 = p1 + c;
        }
```

This code is used in section 11.

13. Output of the statistics is done in a function. This makes it easy to use it for the totals, too. Additionally we must decide here if we know the name of the file we have processed or if it was just *stdin*.

```
⟨ write this file's statistics 13 ⟩ ≡
    wcp(wd, charct, wordct, linect);
    if (argc > 1) {
        printf(" %s\n", argv[i]);
        }
    else printf("\n");
```

This code is used in section 8.

14. Grand totals are just cumulated.

```
⟨ update grand totals 14 ⟩ ≡
    tlinect += linect; twordct += wordct;
    tcharct += charct;
```

This code is used in section 8.

15. Before we stop we have to check if grand total output is needed.

```
⟨ add the grand total line if there were multiple
        files 15 ⟩ ≡
    if (argc > 2) {
        wcp(wd, tcharct, twordct, tlinect);
        printf(" total\n");
        }
```

This code is used in section 6.

16. In this section we declare all the local variables of main. This allows to check how many **register** variables were actually used. The C compiler might ignore some of them if there are too many.

```
⟨ local variables of main 16 ⟩ ≡
    register char *p1, *p2;
    register int c;
    int i, token;
    int status = 0;
    char *wd;
    char b[BUFFERSIZE];
    int f = 0;
```

This code is used in section 6.

17. Printing the output lines. According to the given options we print the values. If an invalid option character was given, we do an emergency stop. But the user is informed about legal parameter settings.

We use a CWEB macro for output of the counts. That makes it easy to show all values in identical format. Of course, a C preprocessor macro would have done the job, too. But that wouldn't have shown the CWEB macro feature.

Because the function is rather short we do not split it into subsections.

```
    define prt_value(#) ≡ printf("%7ld",#);
            break;
⟨ all functions 17 ⟩ ≡
void wcp (wd, charct, wordct, linect)
    char *wd;
    long charct, wordct, linect;
{
    register char *wdp = wd;
    while (*wdp) {
        switch (*wdp++) {
            case 'l': prt_value(linect)
            case 'w': prt_value(wordct)
            case 'c': prt_value(charct)
            default: fprintf(stderr,
            "usage: wc [-clw] [name ...]\n");
                exit(2);
            }
        }
    return ;
}
```

This code is used in section 2.

18. Index. The index is set up by CWEAVE.

Tib: a Reference Setting Package for TEX

J. C. Alexander
University of Maryland

The purpose of this note is to advertise a preprocessor for TEX, called Tib, which creates and sets citations and reference lists.

History

Much of my writing is research papers to be published in professional journals. Each such paper requires a reference list. It was clear to me with the first paper I set in TEX that the error-prone drudgery of creating reference lists should be automated as much as possible. Moreover, I felt, based on several experiences, that any program which created the reference list should be able to set it in a variety of styles and formats (and concomitantly, that users should be able to design new styles). The final impetus was provided by an editor to whom I had submitted a camera-ready manuscript. He responded that the paper was fine, but the publisher specified a different format for the reference list. I reset it, and immediately asked around about bibliography setters on our system, a mainframe running under Unix.

At the time, BIBTEX was known about, but as something that might be available 'someday,' presumably as part of some future version of LATEX. The *troff* program *refer* was available, and there was a successor to *refer*, called *bib*, written by T. A. Budd, which had more features than *refer* and gave the user more control over the output. I decided to spend a week or so modifying *bib*'s output statements to produce TEX output (it would also be a good opportunity for a FORTRAN programmer to learn C). Instead of course, I started modifying features and adding new ones, so that the program evolved over several months into something quite different, with a new name, although retaining the basic ideas of *bib*. A number of colleagues, both on our system and others, expressed interest in the program and I eventually took time to write some documentation. It has been in general use since the beginning of the year, mostly for writing scientific papers and theses, and seems to be useful enough that others in the TEX community might like to try it.

Use

Tib is a preprocessor and does not function as part of a TEX run. Its use can be described schematically as follows:

$$\text{reference file} \xrightarrow{tibdex} \text{index},$$

$$\text{inputfile.tex} + \text{index} \xrightarrow{tib} \text{outputfile.tex}.$$

The reference file is a data base of bibliographic reference items in the format of a *refer* reference file. The index is a sorted list of all words in the reference file. It can be made once and kept. Using the index, Tib can quickly track down a reference item from one or two words. With the use of indices, Tib can handle large reference files. The input file is a TEX file with special symbols where citations are wanted and another symbol where the reference list is wanted. The citation symbols surround words from the reference list such as authors' names or a word from the title. When Tib encounters one of these, it 'looks up' the reference, sets the citation, and later processes the reference items and sets the reference list at the indicated location. The output file is another TEX file with control strings and `\include` statements to completely set the citations and reference list. The special symbols do not interfere with TEX processing, so TEX can be applied to the input document for preliminary proofing and debugging. If necessary, the output file can be edited, for example to eliminate an overfull hbox. Each of the arrows above can be many-to-one. That is, one index can be built from several reference files, and several indices and several input files can be used to create one output file.

There are two other programs in the Tib package besides *tib* and *tibdex*: *tiblist*, which sets all the references in a reference list (e.g. for a *vita*), and *tiblook*, which permits the user to interactively look at entries in a reference file.

Tib is designed to be easy to use. There are some demonstrations provided with the package and emulating them is the best way to get started. On the other hand, Tib is built so that a user can customize it or design an individual style with a straightforward effort.

Features

Some of the features of Tib:

1. Tib is not dependent on any particular macro package. It works with plain TEX files, with LATEX files, with $\mathcal{AMS}$-TEX files. I have not yet tried TEX preprocessors such as Larry Siebenmann's Sweet-TEX (AMS *Notices*, August, 1986), but there should be no conflicts between Tib and such preprocessors.
2. There are two sets of citation symbols, for setting citations in running text or not, and text can be set within a citation.
3. Tib will set citations and reference lists in a variety of styles. For example, citations can be set as reference numbers ('[1]' or '(1)'), as alphabetic codes based on the authors' initials (with or without date), or as last names (also with or without date). Included in the Tib package are several basic styles, including footnotes, as well as the styles of various journals and societies. Included are three AMS styles, two SIAM styles, and ones for IEEE, ACM, APS and AGU journals. A default style is provided, or the user can select the style by a single flag on the program call.
4. Styles are created in a modular fashion, and new ones can be made in a straightforward manner. A user can create his or her own style. In particular, a journal could accept an electronic manuscript processed by Tib and set the manuscript with its own style macros. Of course, the author should have previewed the paper using those macros.
5. Most styles call for processing reference items. Tib can abbreviate first names, reverse first and last names, put names in caps-small caps (if such a font is available) and sort. Such things as TEX's diacritical marks are correctly handled. Tib cam handle some subtleties of English usage. Alphabetizing is done by the first capital letter of the last name. For example, 'deRham' is placed in the R's. Abbreviation is done with some care. 'Heinz-Otto' is abbreviated 'H-O.' 'd'Arcy' is abbreviated 'd'A.'
6. Tib will set up and use 'word-definition' pairs. For example, depending on the style, the 'word' '|UNIV|' in a reference item can be expanded to 'Univ.' or 'University' (or anything else). In particular, *Mathematical Reviews* has provided its complete set of over 1800 journal abbreviations, and Tib can automatically provide the correct abbreviation for these journals. A staff member of my university's library told me that over 60% of inquiries at the information desk concerned incorrect journal abbreviations. Also a list of publishers is provided, set up so that the city of publication is automatically included.
7. The computer's environment can be used to tailor Tib. For example, Tib searches a default index if it cannot find a reference otherwise. The default is a system index, but the user can change this in the environment. Thus for example, the user could list all personal publications in a special file, make an index for it, and it will always be available for Tib. Similarly, the user can change the default style in the environment. If mostly the ACM style is used, it can be made the user's default.
8. A wider variety of fields can be handled than with the original *refer* data bases. For example, for a translated item, both the original and translated versions can be included in one reference item.

Example

This example demonstrates basic Tib use. The following might be an entry in a reference file:

```
%A F. R. Gantmacher
%A M. G. Krein
%I State Press for Technical Literature
%C Moscow-Leningrad
%T Oscillation Matrices and Kernels and
Small Vibrations of Mechanical Systems
%o (Russian)
%D 1950
%t Oszillationmatrizen, Oszillationskerne
und Kleine Schwingungen Mechanischer Systeme
%a Alfred St{\"o}hr
%i Akademie-Verlag
%c Berlin
%d 1960
%O German translation
```

In the input TEX file, this reference might be cited as follows:

```
                ... the theory of Jacobi
matrices.[.krein small.] Furthermore ...
```

Tib is applied to this file, and TEX to the resulting file. In the final printed document, with one of the AMS styles, the line above appears as:

... the theory of Jacobi matrices [Gantmacher and Krein, 1950]. Furthermore ...

The corresponding entry in the reference list is:

F. R. Gantmacher and M. G. Krein [**1950**], *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems* (Russian), State Press for Technical Literature, Moscow-Leningrad, *Oszillationmatrizen, Oszillationskerne und Kleine Schwingungen Mechanischer Systeme*, Akademie-Verlag, Berlin, 1960, German translation.

Portability

The source language of Tib is standard C. It was written on a VAX mainframe operating under Unix, but some attention has been paid to portability. A day was spent with Nelson Beebe at the University of Utah going over changes necessary to bring it up on a non-Unix system. These changes mostly consisted of simplifying file names (over the objections of some Unix devotees), and isolating and documenting system-dependent parts (such as a file sort). Although I have not tried it, I see no particular reason why Tib should not work on a micro. The programs are not particularly big. It might be that large 'word-definition' files should not be used on a micro; the time Tib consumes setting them up might be prohibitive.

Availability

The Tib package of source files, macros and documentation consists of 85 files using about 640K. Included are step-by-step instructions for installation, a list of things non-Unix users should note, a plain TEX source file for a 22 page manual, and some demonstration and test files. The package has been given to Rick Furuta for inclusion with the Unix TEX distribution. It is also available for transporting via *ftp* from eneevax.umd.edu. Login anonymously, change to the subdirectory pub/tib, and copy everything. A third possibility is to get in touch with me about copying a tape. If there is enough interest that it is worth making future versions, they will be announced in the *TUGboat.*

Packed File Format Update

Tomas Rokicki

Some errors in my description of the packed file format and a bug in one of the conversion programs have been brought to my attention in the past year by John Crawford, Wayne Sewell, and others. This note addresses these problems and lists the changes made to the programs. The new, updated programs are on the current distribution tape; they are also available via anonymous FTP from SU-Score.EDU in the directory `<TEX.MF>`.

The first clarification concerns the number of no-op bytes allowed at the end of a packed file. In my original description, I stated that the postamble command was 'followed by just enough no-op commands to make the file a multiple of four bytes long.' For those machines which write binary files in larger blocks, this is inconvenient; thus, I have changed this to allow any number, including zero, of no-op bytes at the end. I have also removed a check for the number of no-op bytes from `pktype`.

The second problem relates to the design size; in the description of the format, I state that the units are $1/2^{16}$ points. Actually, they are in the same units as in the generic font and font metric files: FIXes, or $1/2^{20}$ points.

Finally, in `gftopk`, the minimum bounding box of a character glyph was being calculated incorrectly on the right side. Occasionally an extra row of blank pixels was left in the character. This should not adversely affect any programs; the TFM width and escapement values are still correct, as is the positioning of the reference point. The only manifestation of the problem is in slightly (less than a percent, usually) larger packed files than are absolutely necessary.

All further corrections as listed below pertain to the new version numbers and the change from the old Almost Modern fonts to the new Computer Modern fonts. The following changes should be made in the `WEB` sources rather than the change files, as they are updates and not system dependent fixes.

Changes to **gftopk**:

```
@x
% Version 1.2 fixed one_fourth bug 23 January 1985 TGR.
\def\versiondate{16 August 1985}
@y
% Version 1.2 fixed one_fourth bug 23 January 1986 TGR.
% Version 1.3 fixed bounding box calculations and some documentation.
                                        7 September 1986 TGR
\def\versiondate{7 September 1986}
@z

@x
\font\ninerm=amr9
\let\mc=\ninerm % medium caps for names like SAIL
\font\tenss=amss10 % for 'The METAFONTbook'
@y
\font\ninerm=cmr9
\let\mc=\ninerm % medium caps for names like SAIL
\font\tenss=cmss10 % for 'The METAFONTbook'
@z

@x
  \centerline{(Version 1.2, \versiondate)}
@y
  \centerline{(Version 1.3, \versiondate)}
@z

@x
@d banner=='This is GFtoPK, Version 1.2' {printed when the program starts}
@y
@d banner=='This is GFtoPK, Version 1.3' {printed when the program starts}
@z

@x
followed by just enough |pk_no_op| commands to make the file a multiple
of four bytes long; zero through three are usual, but four are also allowed.
@y
followed by enough |pk_no_op| commands to make the file a multiple
of four bytes long.    Zero through three bytes are usual, but any number
is allowed.
@z

@x
the design size of the file in $1/2^{16}$ points, and the checksum of the
@y
the design size of the file in $1/2^{20}$ points, and the checksum of the
@z

@x
  max_m := min_m + max_m - 1 ;
@y
  max_m := min_m + max_m - 1 - extra ;
@z
```

```
@x
@d preamble_comment == 'GFtoPK 1.2 output'
@y
@d preamble_comment == 'GFtoPK 1.3 output'
@z
```

Changes to pktopx:

```
@x
\def\versiondate{15 August 1985}
%
\font\ninerm=amr9
@y
% Updated documentation, 2.3 version:  7 September 1986
\def\versiondate{7 September 1986}
%
\font\ninerm=cmr9
@z

@x
  \centerline{(Version 2.2, \versiondate)}
@y
  \centerline{(Version 2.3, \versiondate)}
@z

@x
@d banner=='This is PKtoPX, Version 2.2'
@y
@d banner=='This is PKtoPX, Version 2.3'
@z

@x
followed by just enough |pk_no_op| commands to make the file a multiple
of four bytes long; zero through three are usual, but four are also allowed.
@y
followed by enough |pk_no_op| commands to make the file a multiple
of four bytes long.    Zero through three bytes are usual, but any number
is allowed.
@z

@x
the design size of the file in $1/2^{16}$ points, and the checksum of the
@y
the design size of the file in $1/2^{20}$ points, and the checksum of the
@z
```

Changes to pktype:

```
@x
\def\versiondate{16 August 1985}
%
\font\ninerm=amr9
@y
% Documentation updated, 2.2 version, 7 September 1986
\def\versiondate{7 September 1986}
%
\font\ninerm=cmr9
@z

@x
  \centerline{(Version 2.1, \versiondate)}
@y
  \centerline{(Version 2.2, \versiondate)}
@z

@x
@d banner=='This is PKtype, Version 2.1'
@y
@d banner=='This is PKtype, Version 2.2'
@z

@x
followed by just enough |pk_no_op| commands to make the file a multiple
of four bytes long; zero through three are usual, but four are also allowed.
@y
followed by enough |pk_no_op| commands to make the file a multiple
of four bytes long.    Zero through three bytes are usual, but any number
is allowed.
@z

@x
the design size of the file in $1/2^{16}$ points, and the checksum of the
@y
the design size of the file in $1/2^{20}$ points, and the checksum of the
@z

@x
if j > 4 then abort('Too many no-ops at end of file') ;
@y
@z
```

Changes to pxtopk:

```
@x
\def\versiondate{15 August 1985}
%
\font\ninerm=amr9
@y
% Documentation updated (2.3) :  7 September 1986
\def\versiondate{7 September 1986}
%
\font\ninerm=cmr9
@z

@x
  \centerline{(Version 2.2, \versiondate)}
@y
  \centerline{(Version 2.3, \versiondate)}
@z

@x
@d banner=='This is PXtoPK, Version 2.2'
@y
@d banner=='This is PXtoPK, Version 2.3'
@z

@x
followed by just enough |pk_no_op| commands to make the file a multiple
of four bytes long; zero through three are usual, but four are also allowed.
@y
followed by enough |pk_no_op| commands to make the file a multiple
of four bytes long.    Zero through three are usual, but any number
is allowed.
@z

@x
the design size of the file in $1/2^{16}$ points, and the checksum of the
@y
the design size of the file in $1/2^{20}$ points, and the checksum of the
@z

@x
@d preamble_comment == 'PXTOPK 2.2 output'
@y
@d preamble_comment == 'PXTOPK 2.3 output'
@z
```

Hyphenation Exception Log

Below is a list of words that TEX fails to hyphenate properly. It last appeared in Volume 6, No. 3, on page 121. Everything listed there is repeated here.

The first column gives results from TEX's `\showhyphens{...}`; entries in the second column are suitable for inclusion in a `\hyphenation{...}` list.

In most instances, inflected forms are not shown for nouns and verbs; however, all forms should be specified in a `\hyphenation{...}` list if they might occur in your document.

academy	acad-e-my
al-ge-brais-che	al-ge-brai-sche
anal-yse	an-a-lyse
anomaly	anom-aly
anti-nomy(ies)	an-tin-o-my(ies)
ap-pendix	ap-pen-dix
asymp-totic	as-ymp-tot-ic
at-mo-sphere	at-mos-phere
ban-dleader	band-leader
Be-di-enung	Be-die-nung
be-haviour	be-hav-iour
bib-li-ographis-che	bib-li-o-gra-phi-sche
bid-if-fer-en-tial	bi-dif-fer-en-tial
bornolog-i-cal	bor-no-log-i-cal
Brow-n-ian	Brown-ian
buz-zword	buzz-word
cartwheel	cart-wheel
cholesteric	cho-les-teric
database	data-base
dat-a-p-ath	data-path
de-mos	demos
dis-tribute	dis-trib-ute
Di-jk-stra	Dijk-stra
dy-namis-che	dy-na-mi-sche
elec-trome-chan-i-cal	electro-mechan-i-cal
elec-tromechanoa-cous-tic	electro-mechano-acoustic
En-glish	Eng-lish
equiv-ari-ant	equi-vari-ant
Eu-le-rian	Euler-ian
ex-traor-di-nary	ex-tra-or-di-nary
fermions	fermi-ons
flowchart	flow-chart
funk-t-sional	funk-tsional
Gaus-sian	Gauss-ian
ge-o-met-ric	geo-met-ric
gnomon	gno-mon
Greif-swald	Greifs-wald
Grothendieck	Grothen-dieck
Grundlehren	Grund-leh-ren
Hamil-to-nian	Hamil-ton-ian
Her-mi-tian	Her-mit-ian
hex-adec-i-mal	hexa-dec-i-mal
holon-omy	ho-lo-no-my
ho-mo-th-etic	ho-mo-thetic
id-iosyn-crasy	idio-syn-crasy
in-fras-truc-ture	in-fra-struc-ture
Japanese	Japan-ese
jeremi-ads	je-re-mi-ads
Kadomt-sev	Kad-om-tsev
Ko-rteweg	Kor-te-weg
Leg-en-dre	Le-gendre
Le-ices-ter	Leices-ter
Lip-s-chitz(ian)	Lip-schitz(-ian)
macroe-co-nomics	macro-eco-nomics
manuscript	man-u-script
marginal	mar-gin-al
Marko-vian	Mar-kov-ian
Mas-sachusetts	Mass-a-chu-setts
met-a-lan-guage	meta-lan-guage
mi-croe-co-nomics	micro-eco-nomics
mi-crofiche	mi-cro-fiche
mis-ogamy	mi-sog-a-my
mod-elling	mod-el-ling
mo-noen-er-getic	mono-en-er-getic
monopole	mono-pole
monos-pline	mono-spline
monos-trofic	mono-strofic
mul-ti-pli-ca-ble	mul-ti-plic-able
ne-ofields	neo-fields
Noethe-rian	Noe-ther-ian
none-mer-gency	non-emer-gency
nonequiv-ari-ance	non-equi-vari-ance
noneu-clidean	non-euclid-ean
non-i-so-mor-phic	non-iso-mor-phic
nonpseu-do-com-pact	non-pseudo-com-pact
non-s-mooth	non-smooth
No-ord-wi-jk-er-hout	Noord-wijker-hout
parabolic	par-a-bol-ic
parametrized	pa-ram-e-trized
paramil-i-tary	para-mil-i-tary
phe-nomenon	phe-nom-e-non
Poincare	Poin-care
polyene	poly-ene
poly-go-niza-tion	polyg-on-i-za-tion
poroe-las-tic	poro-elas-tic
postam-ble	post-am-ble
Po-ten-tial-gle-ichung	Po-ten-tial-glei-chung
pream-ble	pre-am-ble
pseu-dod-if-fer-en-tial	pseu-do-dif-fer-en-tial
pseud-ofi-nite	pseu-do-fi-nite
pseud-ofinitely	pseu-do-fi-nite-ly
pseud-o-forces	pseu-do-forces

pseu-doword	pseu-do-word
quadrat-ics	qua-drat-ics
quasiequiv-a-lence	qua-si-equiv-a-lence
quasi-hy-ponor-mal	qua-si-hy-po-nor-mal
quasir-ad-i-cal	qua-si-rad-i-cal
quasiresid-ual	qua-si-resid-ual
qua-sis-mooth	qua-si-smooth
qua-sis-ta-tion-ary	qua-si-sta-tion-ary
qu-a-si-tri-an-gu-lar	qua-si-tri-an-gu-lar
re-ar-range-ment	re-arrange-ment
Rie-man-nian	Rie-mann-ian
schedul-ing	sched-ul-ing
Schrodinger	Schro-ding-er
Schwarzschild	Schwarz-schild
semidef-i-nite	semi-def-in-ite
semi-ho-mo-th-etic	semi-ho-mo-thet-ic
ser-vomech-a-nism	ser-vo-mech-anism
setup	set-up
solenoid	so-le-noid
spheroid	spher-oid
stochas-tic	sto-chas-tic
Stokess-che	Stokes-sche
sub-scriber	sub-scrib-er
summable	sum-ma-ble
tech-nis-che	tech-ni-sche
ther-moe-las-tic	ther-mo-elas-tic
times-tamp	time-stamp
ve-r-all-ge-mein-erte	ver-all-ge-mein-erte
Verteilun-gen	Ver-tei-lun-gen
vs-pace	vspace
Wahrschein-lichkeit-s-the-o-rie	Wahr-schein-lich-keits-the-o-rie
waveg-uide	wave-guide
whitesided	white-sided
whites-pace	white-space
Yingy-ong Shuxue Jisuan	Ying-yong Shu-xue Ji-suan

Fonts

Notes on Typeface Protection

Charles Bigelow
Stanford University

Preamble

The main question of typeface protection is: "Is there anything there worth protecting?" To that the answer must certainly be: "Yes." Typeface designs are a form of artistic and intellectual property." To understand this better, it is helpful to look at who designs type, and what the task requires.

Who makes type designs?

Like other artistic forms, type is created by skilled artisans. They may be called type designers, lettering artists, punch-cutters, calligraphers, or related terms, depending on the milieu in which the designer works and the technology used for making the designs or for producing the type.

("Type designer" and "lettering artist" are self-explanatory terms. "Punch-cutter" refers to the traditional craft of cutting the master image of a typographic letter at the actual size on a blank of steel that is then used to make the matrix from which metal type is cast. Punch-cutting is an obsolete though not quite extinct craft. Seeking a link to the tradition, modern makers of digital type sometimes use the anachronistic term "digital punch-cutter". "Calligrapher" means literally "one who makes beautiful marks". The particular marks are usually hand-written letters, though calligraphers may design type, and type designers may do calligraphy.)

It usually takes about seven years of study and practice to become a competent type designer. This seems to be true whether one has a Ph.D. in computer science, a high-school diploma, or no academic degree. The skill is acquired through study of the visual forms and practice in making them. As with geometry, there is no royal road.

The designing of a typeface can require several months to several years. A family of typefaces of four different styles, say roman, italic, bold roman, and bold italic, is a major investment of time and effort. Most type designers work as individuals. A few work in partnership (Times Roman(R), Helvetica(R), and Lucida(R) were all, in different ways, the result of design collaboration). In Japan, the large character sets required for a

typeface containing Kanji, Katakana, and Hiragana induce designers to work in teams of several people.

Although comparisons with other media can only be approximate, a typeface family is an accomplishment on the order of a novel, a feature film screenplay, a computer language design and implementation, a major musical composition, a monumental sculpture, or other artistic or technical endeavors that consume a year or more of intensive creative effort. These other creative activities can be protected by copyright or other forms of intellectual property protection. It is reasonable to protect typefaces in the same way.

The problem of plagiarism

A lack of protection for typeface designs leads to plagiarism, piracy, and related deplorable activities. They are deplorable because they harm a broad range of people beyond the original designers of the type. First, most type plagiarisms are badly done. The plagiarists do not understand the nature of the designs they are imitating, are unwilling to spend the necessary time and effort to do good work, and consequently botch the job. They then try to fob off their junk on unsuspecting users (authors, editors, and readers). Without copyright, the original designer cannot require the reproducer of a type to do a good job of reproduction. Hence, type quality is degraded by unauthorized copying.

Secondly, without protection, designs may be freely imitated; the plagiarist robs the original designer of financial compensation for the work. This discourages creative designers from entering and working in the field. As the needs of typography change (on-line documents and laser printing are examples of technical and conceptual changes) new kinds of typefaces are required. Creative design in response to such needs cannot flourish without some kind of encouragement for the creators. In a capitalist society, the common method is property rights and profit. In a socialist (or, in the past, royalist) society, the state itself might employ type artists. France, as a monarchy and as a republic, has had occasional state sponsorship of typeface design over the past 400 years. The Soviet Union has sponsored the design of new typefaces, not only in the Cyrillic alphabet, but also in the other exotic scripts used by various national groups in the Soviet Union.

Those who would justify plagiarism often claim that the type artists do not usually receive a fair share of royalties anyway, since they have usually sold their designs to some large, exploitive corporation. It is true that type designers, like many artists, are often exploited by their "publishers", but plagiarism exacerbates the problem. Plagiarism deprives the designer of decent revenues because it diverts profits to those who merely copied the designs. Plagiarism gives the manufacturer yet another excuse to reduce the basic royalty or other fee paid for typeface designs; the theme song is that the market determines the value of the design and cheap rip-offs debase the value of a face. For those interested in the economic effects of piracy, it is clear that plagiarism of type designs ultimately hurts individual artists far more than it hurts impersonal corporations.

Kinds of protection for type

There are five main forms of protection for typefaces:

1. Trademark;
2. Copyright;
3. Patent;
4. Trade Secret;
5. Ethics

Trademark. A trademark protects the *name* of a typeface. In the U.S., most trademarks are registered with the U.S. Patent and Trademark Office. The R in a circle $^{(R)}$ after a trademark or tradename indicates U.S. registration. The similarly placed TM indicates that a trademark is claimed, even if not yet officially registered. However, a trademark may be achieved through use and practice, even without registration. Owners of trademarks maintain ownership by use of the trademark and by litigation to prevent infringement or unauthorized use of the trademark by others.

As a few examples of registered typeface trademarks, there are Times Roman (U.S. registration 417,439, October 30, 1945 to Eltra Corporation, now part of Allied); Helvetica (U.S. registration 825,989, March 21, 1967, also to Eltra-Allied), and Lucida (U.S. reg. 1,314,574 to Bigelow & Holmes). Most countries offer trademark registration and protection, and it is common for a typeface name to be registered in many countries. In some cases the registrant may be different than the originator. For example, The Times New Roman (Times Roman) was originally produced by the English Monotype Corporation. In England and Europe, most typographers consider the design to belong to Monotype, but the trademark was registered by Linotype (Eltra-Allied) in the U.S., as noted above.

Trademark protection does not protect the design, only the name. Therefore, a plagiarism of a design is usually christened with a pseudonym

which in some way resembles or suggests the original trademark, without actually infringing on it. Resemblance without infringement can be a fine distinction.

Some pseudonyms for Times Roman are: "English Times", "London", Press Roman, "Tms Rmn". Some for Helvetica are "Helios", "Geneva", "Megaron", "Triumvirate". So far, there seem to be none for Lucida. There are generic typeface classifications used by typographers and type historians to discuss styles, trends, and categories of design. Occasionally these apparently innocuous classification systems are employed by plagiarists to devise generic pseudonyms, such as "Swiss 721" for Helvetica, and "Dutch 801" for Times Roman. It is not certain whether this usage of a generic classification is more for clarification or for obfuscation. In general, the proper tradename is a better indicator of identity, quality, and provenance in typefaces than a generic name. Some people believe that the same is true for other commodities such as wine, where taste is important.

A trademark usually consists of both a proprietary and a generic part. For example, in the name "Lucida Bold Italic", "Lucida" is the proprietary trademark part and "Bold Italic" is the generic part. The generic word "type" is usually understood to be a part of the name, e.g. "Lucida Bold Italic type". Sometimes a firm will append its name or a trademarked abbreviation of it to the typeface name, to achieve a greater degree of proprietary content, e.g. "B&H Lucida Bold Italic".

A related matter is the use of the name of a type's designer. A firm that ethically licenses a typeface will often cite the name of the designer — e.g. Stanley Morison (with Victor Lardent) for Times Roman, Max Miedinger (with Edouard Hoffmann) for Helvetica, Charles Bigelow and Kris Holmes for Lucida. Although a person's name is not usually a registered trademark, there are common law restrictions on its use. The marketing of plagiarized type designs generally omits the names of the designers.

Although Trademark is an incomplete kind of protection, it is used effectively (within its limitations) to prevent the theft of type names. Certain traditional typeface names, usually the surnames of illustrious designers like Garamond, Caslon, Baskerville, Bodoni, and others have become generic names in the public domain. Trademark protection of such names requires the addition of some proprietary word(s), as with these hypothetical creations, "Acme New Garamond", or "Typoluxe Meta-Baskerville".

Copyright. Copyright of typefaces can be divided into two parts: copyright of the design itself; and copyright of the font in which the design is implemented. In the U.S., typeface designs are currently not covered by copyright. This is a result of reluctance by the copyright office to deal with a complex field; by lobbying against copyright by certain manufacturers whose profits were based on typeface plagiarism; by a reluctance of Congress to deal with the complex issues in the recent revision of the copyright law.

The reluctance of Americans to press for typeface copyright may have been influenced by a feeling that typeface plagiarism was good for U.S. high-tech businesses who were inventing new technologies for printing, and plagiarizing types of foreign origin (Europe and England). If the situation becomes reversed, and foreign competition (from Japan, Taiwan, and Korea) threatens to overcome American technological superiority in the laser printer industry, then American firms may do an about-face and seek the protection of typeface copyright to help protect the domestic printer industry. Such a trend may already be seen in the licensing of typeface trademarks by Adobe, Hewlett-Packard, IBM, Imagen, and Xerox in the U.S. laser printer industry.

In Germany, where typeface design has always been a significant part of the cultural heritage, and where typefounding has remained an important business, there are more than one kind of copyright-like protections for typefaces. Certain long-standing industrial design protection laws have been used to protect typeface designs in litigation over royalties and plagiarisms. Further, there is a recent law, the so-called "Schriftzeichengesetz" enacted in 1981, that specifically protects typeface designs. New designs are registered, as is done with copyright in most countries. This law only protects new, original designs. It is available to non-German designers and firms. Therefore, some type firms and designers routinely copyright new designs in West Germany. This gives a degree of protection for products marketed in Germany. Since multinational corporations may find it cheaper to license a design for world-wide use rather than deal with a special case in one country, the German law does encourage licensing on a broader scale than would initially seem to be the case.

France, like Germany, has ratified an international treaty for protection of typefaces. This 1973 Vienna treaty will become international law when four nations ratify it. So far, only France and West Germany have done so, and thus a design must be

protected separately in each country. Even when the treaty becomes law, it will take effect only in those countries that have ratified it. The treaty was principally the work of the late Charles Peignot, a French typefounder, and John Dreyfus, an English typographer and typographic scholar. Presently, typefaces may be registered for protection in France under a 19th century industrial design protection law.

In the U.S., there continues to be some movement for typeface design protection. A proposed bill that would protect the designs of useful articles, like type, has been in committee for a few years. It seems to be going nowhere.

Digital (as opposed to analog) fonts may be protected by copyright of digital data and of computer programs. It has been established that computer software is copyrightable. Therefore, software that embodies a typeface, e.g. a digital font, is presumably also protected. There is some objection to this kind of copyright, on the grounds that the ultimate output of the program or the result of the data (i.e. a typeface design) is not copyrightable. However, the current belief expressed by the National Commission on New Technological Use of Copyrighted Works is that software is copyrightable even if its function is to produce ultimately a non-copyrightable work. Hence, typefaces produced by Metafont or PostScript(R), two computer languages which represent fonts as programs, are presumably copyrightable. Typefaces represented as bit-map data, run-length codes, spline outlines, and other digital data formats, may also be copyrightable. Some firms do copyright digital fonts as digital data.

Note that the designs themselves are still not protected in the U.S. A plagiarist could print out large sized letters (say, one per page) on an Apple LaserWriter, using a copyrighted PostScript digital font, and then redigitize those letters by using a scanner or a font digitizing program and thus produce a new digital font without having copied the *program* or *digital data*, and thus without infringing the copyright on the font. The quality of the imitation font would usually be awful, but it wouldn't violate copyright. Of course, the plagiarist would usually need to rename the font to evade trademark infringement. [As I write these words, I have the guilty feeling that I have just provided a recipe for type rip-off, but others have obviously thought of just such a scheme—John Dvorak has even proposed something like it in one of his columns.]

Design Patent. The designs of typefaces may be patented in the U.S. under existing design patent law. Many designs are patented, but type designers generally don't like the patent process because it is slow, expensive, and uncertain. Nevertheless, some types do get patented, and it is a form of potential protection. Note that this is *Design* Patent—the typeface doesn't have to be a gizmo that does something, it merely has to be unlike any previous typeface. The drawback here is that most attorneys and judges are not aware that there are more than two or three typefaces: say, handwriting, printing, and maybe blackletter. Therefore, litigating against infringement is an educational as well as a legal process. It is easy to see that typeface theft is more subtle than knocking over a liquor store; it may not be illegal and the returns may be greater.

Protections like design patent are available in many other countries, but there is not an international standard (to my knowledge) so the situation must be examined on a country by country basis.

Invention Patent. Methods of rendering typefaces can be patented as mechanical or electronic inventions. For example, the old hot-metal Linotype machinery was protected by various patents, as was the IBM Selectric typewriter and type ball. IBM neglected to trademark the typeface names like Courier and Prestige, so once the patents had lapsed, the names gradually fell into the public domain without IBM doing anything about it (at the time, and for a dozen years or so, IBM was distracted by a major U.S. anti-trust suit). Most students of the type protection field believe that those names are probably unprotectable by now, though IBM could still presumably make a try for it if sufficiently motivated.

There is currently a noteworthy development regarding a patent for outline representation of digital type as arcs and vectors, with special hardware for decoding into rasters. This patent (U.S. 4,029,947, June 14, 1977; reissue 30,679, July 14, 1981) is usually called the Evans & Caswell patent, after its inventors. It was originally assigned to Rockwell, and in 1982, Rockwell sued Allied Linotype for infringement. Allied settled out of court, having paid an amount rumored to be in the millions. Rockwell sold the patent, along with other typographic technology, to Information International, Inc. (III), which then sued Compugraphic for infringement. According to the Seybold Report, a respected typographic industry journal, Compugraphic recently settled out of court for 5 million

dollars. Although many experts believe the patent to be invalid because of several prior inventions similar in concept, it nevertheless seems to be a money-maker in corporate litigation. The Seybold Report has speculated on which firms III would litigate against next. Among the candidates suggested by the Seybolds was Apple for its LaserWriter, which uses outline fonts. Since the entire laser printer industry and the typesetting industry is moving toward outline font representation, Apple is certainly not alone. The Seybolds further speculate on whether the difference between character-by-character CRT typesetting and raster-scan laser typesetting and printing would be legally significant in such a case. Ultimately, some firm will hold out for a court judgement, and the matter will be decided.

Trade Secret. Given that typeface designs have relatively little copyright protection in the U.S., they are often handled as trade secrets. The secret must apply to the digital data or programs only, because the images themselves are ultimately revealed to the public as printed forms. It is much more difficult to reconstruct the formula of Coca-Cola from its taste than it is to reconstruct the design of Helvetica from its look on the page. The exact bitmap or spline outline of a digital font is usually not reconstructable from the printed image, although CRT screen fonts at usual resolutions (60–120 dots per inch) may be reconstructed by patient counting and mapping of bits off a screen display. Typeface licenses often contain stipulations that the digital data will be encrypted and confidential. Just as a firm will protect the secret of a soft drink recipe, so a type firm will protect the exact nature of its digital data.

Ethics. Some typographers are motivated by higher principles than greed, profit, expediency, and personal interest. Idealists afflicted with concepts of ethical behavior and a vision of typography as a noble art may find it distasteful to use plagiarized types. Some graphic designers insist on using typefaces with bona-fide trademarks, both to ensure that the type will be of high quality, and to encourage creativity and ethics in the profession. A consequence of plagiarism that is sometimes overlooked is a general erosion of ethics in an industry. If it is okay to steal typeface designs, then it may be okay to purloin other kinds of data, to falsify one's resume, to misrepresent a product, and so forth. Most professional design organizations attempt to promote ethical standards of professional behavior, and personal standards may extend to avoidance of plagiarism.

The Association Typographique Internationale (ATypI) is an international organization of type designers, type manufacturers, and letterform educators. Its purpose is to promote ethical behavior in the industry, advancement of typographic education, communication among designers, and other lofty aims. Members of ATypI agree to abide by a moral code that restricts plagiarism and other forms of depraved behavior (pertaining to typography). These are noble goals, but some members (especially corporate members) of ATypI, confronted with the pressures and opportunities of commercial reality, nevertheless plagiarize typefaces of fellow members, the moral code notwithstanding. Since ATypI is a voluntary organization, there is very little that can be done about most such plagiarism. Some years back, a world-famous type designer resigned from the ATypI Board of Directors in protest over the organization's flaccid attitude toward plagiarists among its ranks. He has since agreed to sit on the board again, but criticism of the organization's inability to prevent type rip-offs by its own members, not to mention by non-members, continues to be heard. Moderates in ATypI believe that a few morals are better than none. It is not clear whether their philosophical stance derives from Plato, Hobbes, or Rousseau.

Given the general attitude of users toward copyrighted video and software, it is doubtful that ethical considerations will hinder most end-users' attitude to plagiarized type fonts. A desire to have the fashionable "label" or trademark may be a greater motivation toward the use of bona-fide fonts than an ethical consideration.

Further reading

"The State of the Art in Typeface Design Protection", Edward Gottschall, *Visible Language*, Vol. XIX, No. 1, 1985 (a special issue on "The Computer and the Hand in Type Design" — proceedings of a conference held at Stanford University in August, 1983).

Der Schutz Typographischer Schriftzeichen, by Guenter Kelbel. Carl Heymans Verlag KG, Cologne, 1984. (A learned account, in juridical German prose, of the significance of the Vienna Treaty of 1973 and the West German Schriftzeichengesetz of 1981.)

Disclaimer

These notes were originally prepared at the request of Brian Reid, for informal distribution. They are based on the author's review of available literature on the subject of typeface protection, and on personal experience in registering types for trademark, copyright, and patent. However, they are not legal advice. If one is contemplating protecting or plagiarizing a typeface, and seeks legal opinion, it is advisable to consult an attorney. The term "plagiarize" (and words derived from it) is used here in its dictionary sense of "to take and use as one's own the ideas of another" and does not mean that the practice of typeface plagiarism is illegal, as that is determined by the laws of a particular country.

The author is a professor of digital typography as well as a professional designer of original digital typefaces for electronic printers and computer workstations. He therefore has an obvious bias toward the inculcation of ethical standards and the legal protection of artistic property. Other commentators might have a different perspective.

Building Computer Modern fonts

John Sauter

When METAFONT version 1.0 was released I eagerly obtained a copy, because I wanted to use the Computer Modern fonts on my DEC LN03. By experiment, and with some assistance from Professor Knuth, I determined the METAFONT device-dependent parameters for the LN03. The non-obvious parameters are: *blacker* 0.65, *fillin* −0.01, *o_correction* 0.5.

I then found that the process of building all of the font files on a VMS system, while straightforward, is not trivial. I have written some VMS command procedures which build the font files, and I would be happy to share them with the TEX community. They require about $2\frac{1}{2}$ days of CPU time on a VAX-11/785, so I would also be willing to share the results of this process.

Of course, I was not satisfied with just the 75 standard fonts in the standard 7 magnifications. I also wanted Computer Modern Symbols in 12-point, since I use 12-point a lot due to the low resolution of the LN03. In addition, I like to use Computer Modern Sans Serif for acronyms, but CMSS12 looks a little too large, so I wanted Computer Modern Sans Serif 11-point. I could have gotten these using magnified fonts, of course, but that doesn't seem right: I wanted fonts that were designed for the size in which I was using them. I expected to want more fonts than these someday, like a 20-point monospaced font, so I wanted a way to build non-standard Computer Modern fonts in a more-or-less automatic way.

My solution to this problem was to create alternative parameter files to produce the Computer Modern fonts. By ignoring point size I counted 31 Computer Modern font families: CMB, CMBSY, CMBX, CMBXSL, CMBXTI, CMCSC, CMDUNH, CMEX, CMFF, CMFI, CMFIB, CMINCH, CMITT, CMMI, CMMIB, CMR, CMSL, CMSLTT, CMSS, CMSSBX, CMSSDC, CMSSI, CMSSQ, CMSSQI, CMSY, CMTCSC, CMTEX, CMTI, CMTT, CMU and CMVTT. I created the corresponding 31 parameter files, each of which takes a point size as input. If you give the alternative parameter file a point size which corresponds to one of the standard Computer Modern fonts, it produces exactly the same results as the standard parameter file. If you give it a point size which does not correspond to a standard font, it interpolates or extrapolates each of the font parameters to produce what seems to me to be a reasonable value, based on all the standard values for that parameter in that font family. In font families in which only one point size is given, such as CMFF10, I couldn't do more than linear extrapolation. In most of the families, though, I was able to write an algorithm for each parameter which gave reasonable results for all point sizes between 5 and 100. Sometimes the formulae are quite complex, in order to exactly match the standard values at the standard point sizes.

Wherever possible I tried to use common files for similar calculations. For example, the upper case part of CMCSC10 is almost identical to CMR10, so I used a single file, `COMPUTE_CMR`, to compute the common parts of both. Thus, even though CMCSC is given only in 10-point, I can use the algorithms of CMR to give better values than I could have gotten through linear extrapolation. Similarly, CMSY is very similar to CMMI, so I can use the CMMI calculations to give a better CMSY12.

To guard against typographical error, I have used these alternative parameter files to create all of the standard point sizes and magnifications for the LN03, and compared them with the files produced by the standard parameter files. The `.TFM` files had to match exactly, or the fonts could not be called

Computer Modern. I was willing to accept some variation in the pixel files, but as it turned out after I removed all the errors from the parameter files there were no differences in the pixel files either.

I had some trouble matching the .TFM files because of roundoff errors, particularly in fonts like CMCSC, where some parameters are computed by modifying others. I solved this by using units of $\frac{1}{360}$ of a point rather than the standard $\frac{1}{36}$ of a point when I could do so without overflow, and by adding "fudge factors" to correct the remaining small differences.

I would like to distribute these files through the normal mechanism from Stanford, since I have received very good service from Maria Code, but I don't know how. I intended to research this problem before writing this article, but the deadline for this issue snuck up on me so I won't have time. Therefore, if anybody wants VMS Backup copies of all that I have described above—command files, alternative parameter files, and the resulting .TFM and pixel files for the DEC LN03—just write me and I'll send you a magnetic tape by return mail. If you can't read 6250 BPI tapes be sure to let me know, since that is my default density: it lets me use a smaller tape.

Editor's note: Arrangements are being made to include at least some of the files described above on the VAX/VMS distributions from Stanford and from Kellerman & Smith. The files have also been offered for inclusion on the VAX/Unix tape; this may take a bit more time to effect, since the algorithms must be translated to a Unix shell script, and VMS dependencies removed. Anyone wishing to volunteer to undertake a translation to Unix should communicate with Pierre MacKay or Barbara Beeton.

Output Devices

TEX and Macintosh – New Directions in Preview

Rick Jansen
Academic Computing Services Amsterdam (SARA)

At SARA [1], the Academic Computing Services Amsterdam, the Macintosh application TEX Preview has been developed, a tool that you can use to view DVI files with a Macintosh microcomputer directly after running TEX on the host computer.

Why Preview

TEX is not an easy to use word processor, in fact TEX commands are quite error prone, so "debugging" a TEX document can be a rather tedious process. This is especially true if the DVI files are printed with an off-line typesetting machine. At SARA it may take as much as three days to get your DVI file printed. Clearly, it is very frustrating for the user to get (expensive) output with some typing errors or an entirely italic paragraph because there was a "}" missing.

Previewing of documents before actually typesetting them can be very useful to correct errors and to try things out. It saves you time, money and a lot of frustration if you can get a view of the results directly after running TEX.

Why Macintosh

To represent pages formatted by TEX with different fonts and styles you need a graphics device with very powerful graphics. As speed and cost are the main considerations, a graphics terminal and some kind of driver program on the host computer do not suffice. The previewing device must have these capabilities of its own. Therefore the Apple Macintosh was chosen for a TEX Preview facility. Macintosh is a microcomputer with very powerful graphics, different fonts, fontsizes, styles, etc. Its excellent datacommunication facilities enable you to easily connect Macintosh to the host computer running TEX. You can use Macintosh as a terminal for editing the TEX input and for transferring the DVI file from the host computer to a Macintosh diskette for previewing.

TEX Preview

The goal of TEX Preview is not to be a substitute for a typesetter for printing the final copy. TEX Preview is a *tool* that you can use to display pages from a DVI file in a quite recognizable way. With TEX Preview you can:

- Check results directly after running TEX;
- See what an entire page looks like, its general appearance;
- View an enlarged part of the page, for viewing a formula for example;
- Check stylistic variations like italic, bold, slanted etc. and mathematical symbols.

As with most Macintosh applications, the operation of TEX Preview almost guides itself. Due to the consistent and intuitive Macintosh user interface you will know how to use TEX Preview if you know how to use any other Macintosh program.

TEX Preview displays information on screen in so called "windows". A window is a rectangular part of the screen where a program can display text or graphics. Windows can overlap each other, but always there is one window the "frontmost" or "active" window. The advantage of using windows on a screen is that the total area to display information in is much larger than the area of the physical screen. Also, the information on screen can be logically grouped into specific windows. A window that is overlapped by another window, as in Figure 1, can be made the frontmost by pointing at it with the mouse and pressing the mouse button, see Figure 2.

The windows on screen

In TEX Preview there are three windows for on-screen viewing of pages. With these windows you can:

- view entire pages on the Macintosh screen to get an overall impression of the layout of a page (Overview window);
- view an enlarged section of a page, for example to check a single formula or paragraph (Zoom window);

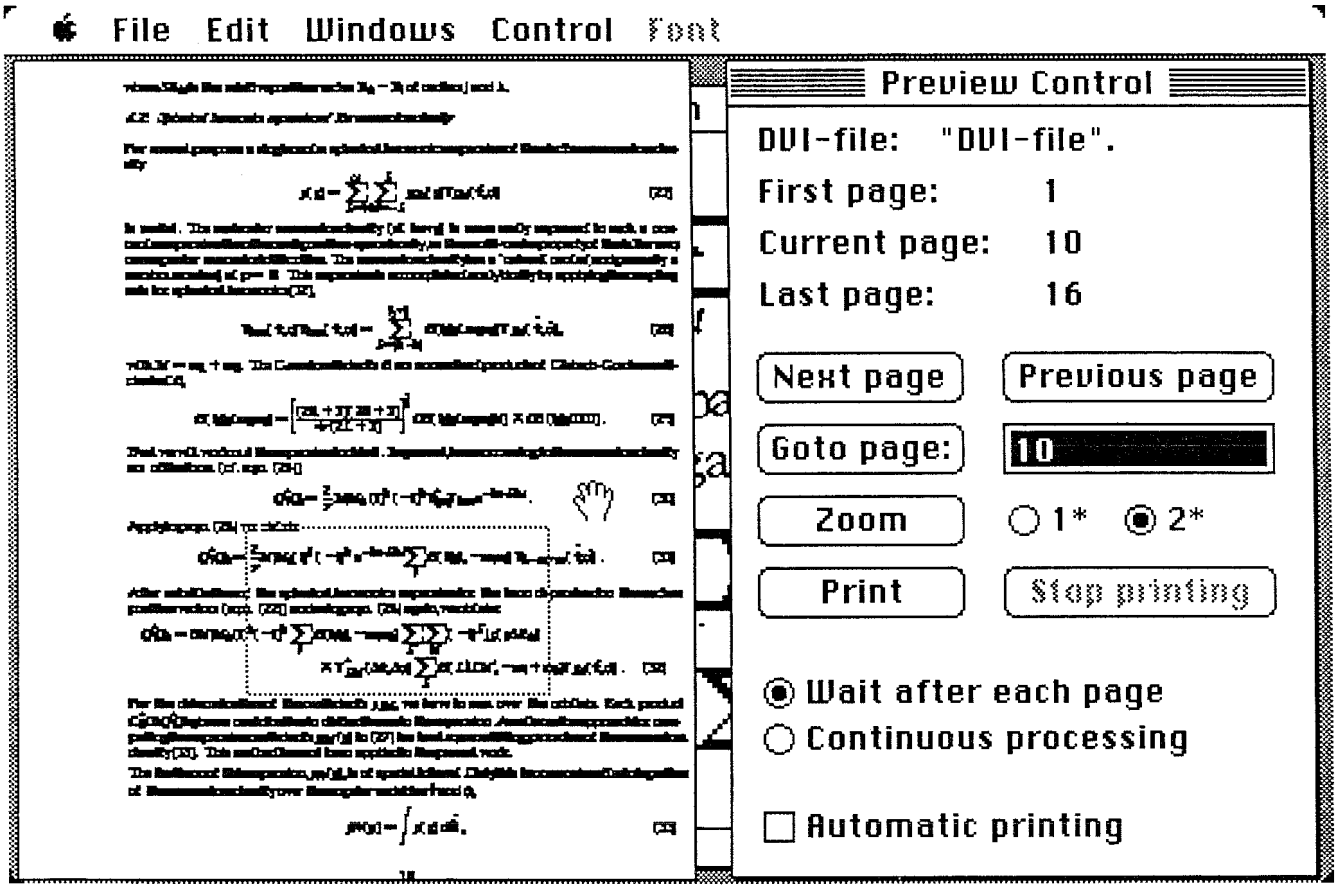


Figure 1. The Zoom window is the behindmost window

- issue commands by pressing buttons with the mouse (Preview Control window).

Overview

The Overview window displays the entire page. The page is reduced in size to fit on screen, see the leftmost window in Figure 1. The Overview window gives you an impression of the general layout of a page.

When the mouse is in the Overview window it changes from the arrow into a hand and a dotted rectangle appears. You can move this dotted rectangle with the mouse. The area inside the dotted rectangle can be viewed enlarged in the Zoom window. This process may remind you of "Show Page" in MacPaint, another Macintosh program.

Zoom

The Zoom window (see Figure 2) displays part of the page at its actual size, or at a size twice the actual size. This is useful to get a close view of a formula or paragraph. The Zoom window has two so called scrollbars with which you can scroll the window over the entire page. When TEX Preview is just started the Zoom window lies behind the other windows (see Figure 1). To view part of the page enlarged you can get the Zoom window to the front, as is shown in Figure 2.

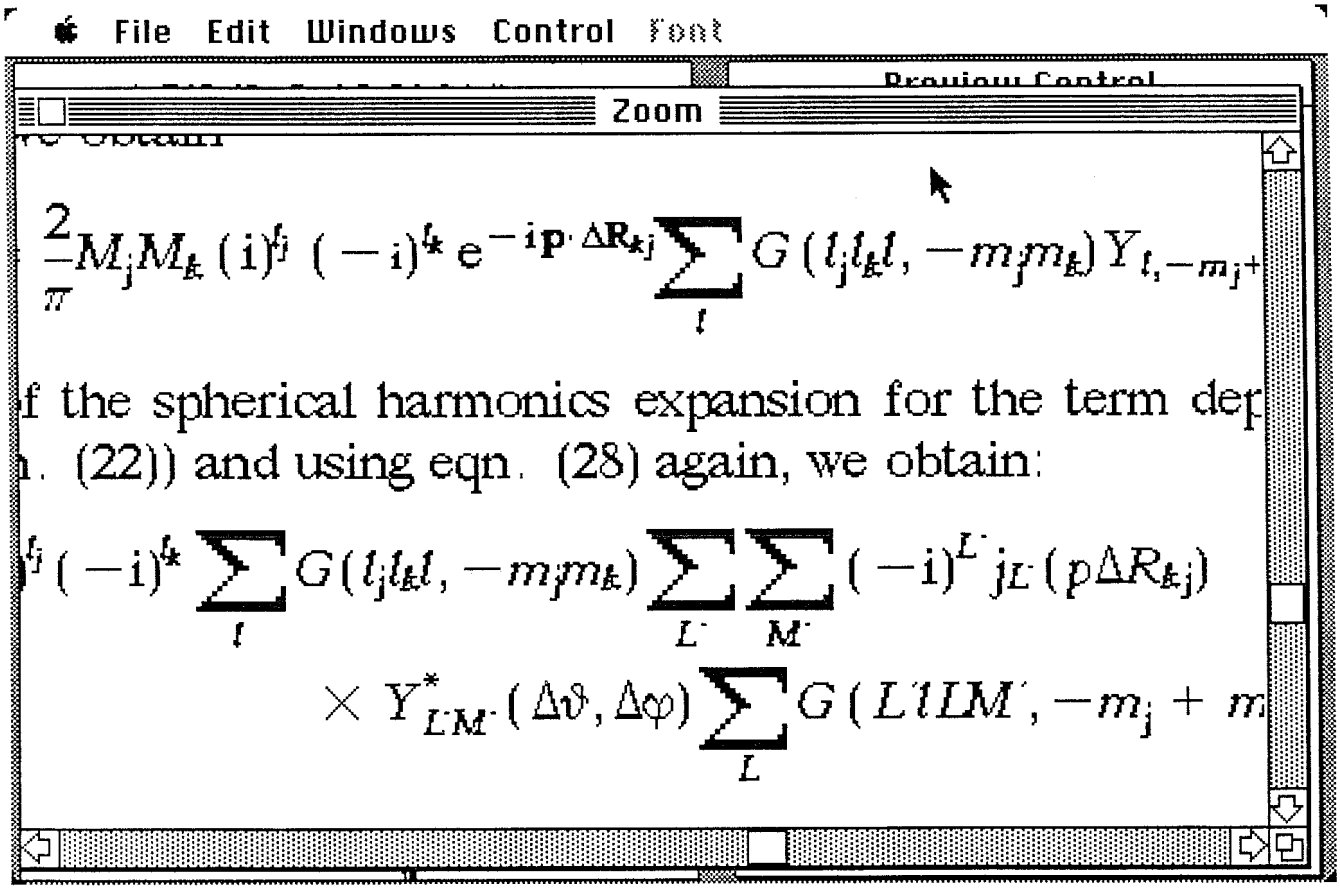


Figure 2. The Zoom window is now the frontmost

Preview Control

With the Preview Control window, the rightmost window in Figure 1, you control the operation of TEX Preview. There are buttons to give commands and there are other controls to select certain options. The window also contains information on the current situation, for example the first, current and last page number etc. With the mouse you can "press" a button and the command with which the button is labeled is performed. For example if you "click" the "Next page" button TEX Preview will display the next page from the DVI file.

About fonts

TEX formats text for a specific output device. This output device can be a photo typesetter or for instance a laserprinter. A typesetter or laserprinter prints text using its built–in fonts.

TEX knows of the output device the dimensions of each character within each font. As these fonts are somewhat different for each typesetter, the DVI file which TEX creates contains device *dependent* information, or better said *font* dependent information. Only the format of the DVI file is device independent.

The names of the fonts TEX used when formatting the text are stored in the DVI file, so the program which actually prints the DVI file on a typesetter or laserprinter knows which fonts to use. This is, of course, also the case for TEX Preview. To be able to generate typesetter–like output TEX Preview must have access to typesetter–like Macintosh fonts.

TEX Preview reads the DVI file and decides which Macintosh font to use for the font used by TEX. For this purpose a "font mapping" has been implemented in TEX Preview. The font mapping tells TEX Preview which Macintosh font to use for a font used by TEX. You may have to modify the font mapping to get TEX Preview use the corresponding fonts.

TEX Preview has the following fonts built–in:

cmr-7	Times Roman like font, 7pt
cmr-10	Times Roman like font, 10pt
cmti-10	Times Roman Italic like font, 10pt
cmsy-10	Mathematical symbol font, 10pt
cmmi-10	Mathematical symbol font, 10pt
cmex-10	Mathematical symbol extension font, 10pt

If one of these fonts is requested at another size the font is scaled to that size from an existing size.

Printing

Besides viewing pages from the DVI file on screen you can also print pages with the ImageWriter matrix printer or with the LaserWriter laser printer. Printing is one of the best ways of previewing, as you can view the entire page at its actual size, and at a resolution much higher than the resolution of the screen. The only disadvantage is that printing is much slower than viewing pages on the screen. You can print one page at a time, print the entire DVI file, or you can specify a range of pages to be printed.

For previewing DVI files from an existing TEX the ImageWriter matrix printer is the best choice as you can print the different fonts with the ImageWriter just like you can see them on the screen. The LaserWriter is less suited for previewing as you cannot alter the LaserWriter fonts. Also the LaserWriter fonts have a different layout than the TEX fonts, this is especially the case for the math fonts.

Printing with the ImageWriter offers the highest resolution available in a stand alone Macintosh system. The ImageWriter has a high resolution mode, which is used when high quality printing, of 144 pixels per inch. The Macintosh screen has half this resolution: 72 pixels per inch. At a resolution of 144 pixels per inch you can make a reasonably good resemblance of the fonts of your typesetter. Figures 3 and 5 are sample pages from the typesetter (a Harris 7500), Figures 4 and 6 are the corresponding pages printed with TEX Preview and the ImageWriter matrix printer. [Editor's note: Figures 3–6 have been reduced to 70%.]

Using the LaserWriter

TEX formats your text with knowledge of the dimensions of the fonts of the printing device. The LaserWriter has, at this moment, four fonts stored in the LaserWriter in ROM: Times, Helvetica, Courier and a Symbol font. So, to take full advantage of the LaserWriter's capabilities, you will have to feed TEX with the font definitions of these LaserWriter fonts for proper formatting. For previewing you can, of course, use the LaserWriter, but special symbols like integrals and sum signs will not be printed correctly without adapting your TEX.

Configuration

TEX Preview will work with 128K, 512K and Macintosh/XL systems. However, printing with a 128K system is very slow. If you want to print pages a 512K Macintosh is recommended.

Apart from the fundamental approximations, the method is implemented with a very efficient computational scheme. First, the Diophantine integration method [25] is used for the calculation of the Fock-matrix elements. Second, a single centered (at all nuclei) charge density fit is applied for the calculation of both Coulomb and Exchange potentials in the Fock operator. This implies a more complicated use of a computer program package applying the HFS method compared to the (standard) Hartree-Fock program packages.

In the HFS package point group symmetry is applied for the construction of atomic or molecular orbitals from the primitive basisfunction sets. Additionally a molecular wavefunction can be built up from fragment orbitals rather than from basisfunctions. Although these two ways of describing the wavefunction are numerically equivalent, the analysis of the molecular orbitals will be more straightforward if this is done in terms of significant fragment orbitals. Throughout this study computation of molecular orbitals will be performed in a basis of fragment orbitals. Such a basisset can be any truncated set of the fragment orbitals, provided that the occupied orbitals are available. If no virtual fragment orbitals are included in the overall molecular basis we speak of a 'minimal-basis' wavefunction, which is assumed to be more suitable for interpretation than the one obtained by using a single primitive basisfunction for each atomic orbital. The latter wavefunction we will refer to as a single-ζ wavefunction.

As already noted above, the basissets always consist of Slater type functions. This might be particularly important for the momentum-space electron density discussed in later chapters: the position and momentum space representations are inversely weighted, a diffuse orbital in one representation is contracted in the other representation and vice versa. Because the wavefunctions are always evaluated in the position-space representation of the Schrodinger equation, the amplitude at high r is important for the amplitude at low p, being of primary importance in momentum space studies.

In many parts of this work the bond energy plays an important role. This energy is always calculated with the Transition-State method as developed by Ziegler and Rauk [26], which has been implemented along with the HFS program.

5.3. Practical use and some results

The topics discussed in the previous section are of a methodological nature but do have some implications on the practical use of the HFS computer program package. Generally molecular orbital calculations within the HFS scheme are performed within the so-called frozen core orbital approximation. Hence a basisset is extended with one STO for each core orbital to provide orthogonalization among core and valence orbitals. So three basissets are needed for most atoms, viz. the valence set, the core set (including expansion coefficients for the core orbitals) and a fitfunction set. By now the core sets are standardized and optimized valence sets are available for all atoms [27] in single-ζ, double-ζ and triple-ζ quality. The choice of fitfunctions is less obvious. In most cases a selection is made out of basisfunction products using criteria of overlap (in order to avoid near-dependency) and the position of the maxima in the fitfunctions. Recently new investigations have been initiated to obtain a better standard and general quality of automatically generated (even tempered) fitfunction sets. The core-potentials are computed with a numerical Xα method and must be available for each atom during a run of the program. The numerical integration method requires three user specified parameters for each atom in the calculation: the halfwidth and an exponent for the distribution sphere, and the fraction of the total number of integration points for the molecule. The integration method introduces a not exactly known amount of numerical noise which may become evident under different circumstances, such as the influence of the initial guess wavefunction, or the route along which the bond energy is computed. These effects may be caused by e.g. local minima in the SCF surface or errors in potentials due to the electron density fit. A very careful study

Figure 4. The same page from the ImageWriter (70%)

where $\Delta\mathbf{R}_{kj}$ is the relative position vector $\mathbf{R}_k - \mathbf{R}_j$ of centers j and k.

6.2. Spherical harmonics expansion of the momentum density

For several purposes a single center spherical harmonics expansion of the electron momentum density

$$\rho(\mathbf{p}) = \sum_{L=0}^{\infty} \sum_{M=-L}^{L} \rho_{LM}(p) Y_{LM}(\hat{\vartheta}, \hat{\varphi}) \tag{27}$$

is useful. The molecular momentum density (at low p) is more easily expressed in such a one-centre expansion than the configuration-space density, as the multi-centre property of the latter may cause greater numerical difficulties. The momentum density has a 'natural' centre (and generally a maximum value) at $\mathbf{p} = \mathbf{0}$. This expansion is accomplished analytically by applying the coupling rule for spherical harmonics [32],

$$Y_{l_1 m_1}(\vartheta, \varphi) Y_{l_2 m_2}(\vartheta, \varphi) = \sum_{L=|l_1-l_2|}^{l_1+l_2} G(l_1 l_2 L m_1 m_2)\, Y_{LM}(\hat{\vartheta}, \hat{\varphi}), \tag{28}$$

with $M = m_1 + m_2$. The Gaunt coefficients G are normalized products of Clebsch-Gordan coefficients CG,

$$G(l_1 l_2 L m_1 m_2) = \left[\frac{(2l_1+1)(2l_2+1)}{4\pi(2L+1)}\right]^{\frac{1}{2}} CG(l_1 l_2 L m_1 m_2 M) \times CG(l_1 l_2 L 000)\,. \tag{29}$$

Next we will work out the expansion in detail. In general, terms occurring in the momentum density are of the form (cf. eqn. (26))

$$Q_j^* Q_k = \frac{2}{\pi} M_j M_k\, (\mathrm{i})^{l_j} (-\mathrm{i})^{l_k} Y_{l_j m_j}^* Y_{l_k m_k} \mathrm{e}^{-\mathrm{i}\mathbf{p}\cdot\Delta\mathbf{R}_{kj}}\,. \tag{30}$$

Applying eqn. (28) we obtain

$$Q_j^* Q_k = \frac{2}{\pi} M_j M_k (\mathrm{i})^{l_j} (-\mathrm{i})^{l_k} \mathrm{e}^{-\mathrm{i}\mathbf{p}\cdot\Delta\mathbf{R}_{kj}} \sum_{l} G(l_j l_k l, -m_j m_k)\, Y_{l,-m_j+m_k}(\hat{\vartheta}, \hat{\varphi})\,. \tag{31}$$

After substitution of the spherical harmonics expansion for the term dependend on the nuclear position vectors (eqn. (22)) and using eqn. (28) again, we obtain:

$$\begin{aligned} Q_j^* Q_k = 8 M_j M_k (\mathrm{i})^{l_j} (-\mathrm{i})^{l_k} \sum_{l} G(l_j l_k l, -m_j m_k) \sum_{L'} \sum_{M'} (-\mathrm{i})^{L'} j_{L'}(p\Delta R_{kj}) \\ \times Y_{L'M'}^*(\Delta\vartheta, \Delta\varphi) \sum_{L} G(L'lLM', -m_j + m_k) Y_{LM}(\hat{\vartheta}, \hat{\varphi})\,. \end{aligned} \tag{32}$$

For the determination of the coefficients ρ_{LM}, we have to sum over the orbitals. Each product $C_{ij}^* C_{ik} Q_j^* Q_k$ gives a contribution to distinct terms in the expansion. An alternative approach for computing the expansion coefficients $\rho_{LM}(p)$ in (27) is a least squares fitting procedure of the momentum density [33]. This method has not been applied in the present work.

The first term of this expansion, $\rho_{00}(p)$, is of special interest. Only this term remains after integration of the momentum density over the angular variables $\hat{\vartheta}$ and $\hat{\varphi}$,

$$\rho_{00}(p) = \int \rho(\mathbf{p})\, \mathrm{d}\hat{\Omega}, \tag{33}$$

Figure 5. Another sample page from the Harris 7500 typesetter (70%)

Down–loading DVI files

For previewing of a DVI file this DVI file has to be transferred from the host computer to a Macintosh diskette. TEX Preview reads a DVI file from the diskette and displays the pages.

There is a large number of very good terminal emulators available for Macintosh. Most of these packages also have a file transfer facility built–in, which makes it very easy to transfer the DVI files from the host computer to Macintosh. Two common terminal emulators that offer error free file transfer with the popular XMODEM protocol are VersaTerm [2] and MacTerminal [3]. Also, there are special programs for file tranfer. A very common program is called Kermit [4].

Both XMODEM and Kermit use an error correcting protocol to ensure that errors during the file tranfer are detected and corrected. For file transfer with an error correcting protocol you always need two programs, one on the computer where the to be transferred file is, and one on the computer the file is to be transferred to. These two programs "talk" to each other, one program sends the data, the other receives the data. The sending program adds some extra information so that the receiving program can detect errors in the transfer. If an error occurs the receiving program "asks" the sending program to retransmit the part where the error occurred.

The XMODEM protocol is implemented in many terminal emulators. On the host computer there must also be a program to support the XMODEM protocol. For UNIX systems these are two public domain programs called MacPut and MacGet [5] (both written in "C"). Kermit is also available for a wide range of machines (from microcomputers to mainframes), including Macintosh. Kermit for Macintosh is a stand alone program with a VT100 terminal emulator built–in.

Availability of TEX Preview

TEX Preview is available in two versions: for TEX–80 and TEX–82. The package is directly available from SARA, at a price of Dfl 300 (which is ca US$ 110 at the current exchange rate). It includes one diskette and a user's guide. The disk contains the TEX Preview program and some utility programs. With one of these utilities, called "TeXSetScrap", you can put Macintosh pictures created with for example MacPaint or MacDraw into your TEX document, using *standard* TEX. (No \special command needed.)

If you are interested in obtaining the program then please contact us at one of the addresses listed below. We will then send you an order form.

References

1 SARA is the computing centre of the University of Amsterdam (UvA), the Free University (VU) and the Centre for Mathematics (CWI, formerly called MC). SARA is a foundation maintaining computing facilities for its founders. SARA's computing facilities include:

- General service on two Cyber 170/750 mainframes;
- Supercomputer service on a Cyber 205;
- Front–end service for the Cyber 205 on a Cyber 825;
- IBM 4381 service for the universities administrations;
- Electronic mail services (EARN/BitNet) on a VAX 750.

Address:
SARA,
Kruislaan 415,
1098 SJ Amsterdam,
The Netherlands.
Mail address:
SARA,
P. O. Box 4613,
1009 AP Amsterdam,
The Netherlands.
Telephone: (31)–20–5923000.
Telex: 12571 mactr nl. (Be sure to mention SARA in the Telex.)
E–mail: Bitnet: Rick@HASARA5

2 VersaTerm supports VT100, Tektronix 4010 and Data General D200 terminals. It is distributed by Peripherals Computers & Supplies, Inc., 2232 Perkiomen Avenue, Mt. Penn, PA 19606.

3 MacTerminal is available from Apple dealers.

4 Kermit was developed by Frank da Cruz at Columbia University. Kermit is available through user groups or directly from Columbia University.

5 MacPut and MacGet were written in "C" by Dave Johson at Brown, and have been maintained (at New York University) by David Spector.

Crudetype
An Adaptable Device Driver

R.M. Damerell
Royal Holloway and Bedford College

Introduction

The purpose of this program is to provide a framework for users to write TEX device drivers for a variety of 'crude' devices. Roughly speaking, 'crude' means any printer that cannot print the fonts that METAFONT generates. This would include daisy-wheels and most impact dot-matrix printers. Considered as output printers for TEX, such devices usually have some of the following defects:

1. Coarse resolution.
2. Restricted character set.
3. Some printers cannot do reverse line feeds; some can, and tear the paper.
4. Slow interface between CPU and printer.

Although such printers cannot do justice to TEX output, drivers for them are still needed. Some users cannot afford high quality printers. Some can only afford to use them for final output; so they need to make proofs on a cheaper printer. Also, anybody who has a high quality printer may well need to refer to various WEB files while writing a driver for it. These can become illegible in critical places. Figure 1 gives a sample from DVItype.

Using the basic (line printer) version of Crudetype, we can get a copy of these formulae which is at least legible, even though the result may not be at all pleasant to look at. A further difficulty with conventional drivers is that most of these use the algorithm 'paint a page of pixels, send it down the line'. This places a heavy load on both the host computer and the link to the printer. Of course, one can try to reduce this load by various optimisations, (e.g. by writing critical bits of code in machine language) but this makes the program non-portable, and often introduces bugs. Crudetype is written entirely in PASCAL, without any attempt at optimisation. When compiled on a VAX 780 with the NO-OPTIMISE, CHECK and DEBUG qualifiers it runs at about 2–3 seconds a page. These times are highly variable, and the VMS optimiser reduces them by about 10–15%.

Printers vary enormously both in their capabilities and in the commands that drive them. The behaviour of Crudetype is controlled by a large number of constants, which supposedly describe how the target printer does things. This does have the disadvantage that the user must compile a separate copy of the program for each different printer, and also devise some way to ensure that he uses the right version for the intended printer. But the only alternative seemed to be that Crudetype should read and parse a file describing the printer and this appeared to be unbearably messy. Ideally, these constants should be so designed that:

1. Any decent printer can be driven by assigning the right values to these constants and recompiling.
2. If the printer is properly documented, it should be immediately obvious what are the correct values for all these constants.

At present I do not have enough experience of different printers to come near this ideal. In particular, some printers can download characters. The problems of writing a program to support this facility in proper generality are horrible and ghastly. I have not made any serious attempt yet to tackle them. There are just a few places where a hook appears, and I hope eventually to attach actual routines for downloading.

Some of the more obvious problems of downloading are: when can you download? any time? start of page? or only at start of document? Can you load one character, or must you load a whole font at a time? How much memory does the printer provide for downloading? How efficiently does it use its memory? What does it do when it runs out? Can you clear out old fonts to make more space? What is the format of a download command? What parameters does it need, in what order, with what punctuation? In what order must pixels be sent? Should they be compressed, and how?

Implementation

This program was originally based on D.E.Knuth's program DVItype, but so many changes were needed for various reasons that there is not much of the original code left. The original version of Crudetype was aimed at a line printer (because everybody has

```
A \\{fix\_word} whose respective bytes are $(a,b,c,d)$ represents the number
$$x=\left\{\vcenter{\halign{$#$,\hfil\qquad&if $#$\hfil\cr
b\cdot2^{-4}+c\cdot2^{-12}+d\cdot2^{-20}&a=0;\cr
-16+b\cdot2^{-4}+c\cdot2^{-12}+d\cdot2^{-20}&a=255.\cr}}\right.$$
```

Figure 1. DVItype output can become illegible in critical places.

these), and was written on the VAX/VMS operating system. It is intended to be easily adaptable both to other systems and to other printers. So most of it is written in standard PASCAL. (It is not possible to tell exactly how much of it is standard, as we do not have a certified compiler.) But in some places, it is necessary to use extensions. In particular, `Crudetype` must read the font files, whose names are dynamically specified. That would be impossible in pure PASCAL.

`Crudetype` also uses non-standard code in order to talk to the user's terminal. It asks for the name of the `DVI` file, and for the first page and the number of pages to print. If an operating system forbids terminal interaction, the installer will have to find another way to give the program this information. As file handling is inevitably system-dependent, I have here allowed myself a lot of latitude in using VMS-specific procedures. If `Crudetype` cannot find a file, it will ask the user for another name. On the other hand, all files are read and written sequentially, and I have got rid of all uses of the default **case** statement. The intention is that all the system-dependent stuff goes near the top of the file, and all printer-dependent stuff at the end. Then with any luck you can merely concatenate Change files for the local system and the local printer, instead of having to merge them. All the code that is known to be non-standard has been carefully segregated from the rest of the program. It amounts to about 20 lines out of 750.

It is clearly impossible to predict what difficulties will appear in trying to install `Crudetype` on other systems. It would seem to be advisable to get the line printer version working before trying to adapt it for any other printer. To try to ease the process, I propose to distribute some test files with the program; each of these will come with the corresponding `DVI` file and (lineprinter) output file. I have also written a change file for a Phillips printer; but it should be understood that this file only works on a particular model of Phillips GP300, with a particular suite of resident fonts. It is only intended as a pattern to show what a printer change file should look like.

Translating the device-independent file

This part of `DVItype` is long and complicated, and I have tried to tidy it up, using ideas originally due to Prof. M.Doob. `DVItype` has seven functions for reading integers from the `DVI` file and two more for the `TFM` file. By passing suitable parameters, these have been reduced to three. `DVItype` processes each `DVI` command via a very big **case** statement. This can be rewritten in a much more readable form. To begin with, 192 of these cases are very similar, so let's get rid of them first:

```
⟨ Get DVI command and execute it 7 ⟩ ≡
com ← get_byte(dvi);
if com < 128 then
  begin set_character(com); move_right;
  end
else if (com ≥ 171) ∧ (com ≤ 234) then
    change_font(com − 171);
  else
```

See also section 8.

Now we come to the **case** statement proper. The macro *four_cases* generates 4 case labels, and generates a procedure call that reads a parameter from the `DVI` file and assigns it to *par*. A similar macro is needed for the movement commands; it has to construct a signed parameter.

```
⟨ Get DVI command and execute it 7 ⟩ +≡
case com of
  four_cases(128)(set_character(par);
      move_right; );
132: begin set_rule; move_right;
  end;
  four_cases(133)(set_character(par));
137: set_rule;
138: do_nothing;
139, 247, 248, 249: bad_dvi('byte:␣', com : 1,
      '␣out␣of␣context␣inside␣page');
140: end_page ← true;
141: push;
142: pop;
  move_cases(143)(h ← h + par);
147:  { W0 }
  h ← h + w;
  move_cases(148)(w ← par; h ← h + w);
  ⟨ about 15–20 lines omitted here 0 ⟩
end;
```

Because all the cases are thus collected together, it is now very easy to check that the **case** statement has 64 labels in the subranges 128 .. 170 and 235 .. 255. Therefore all 256 possible values of *com* produce a defined action; so we can correctly omit the default **case** statement. The resulting code is not quite as beautiful as this example suggests. When `Crudetype` is scanning through the file and looking for the first page to be printed, it must discard `DVI` parameters instead of using them, and so a second **if** statement and **case** statement are needed. But `DVItype` also has a second **case** statement (in function *first_par*), so I maintain that this presentation is still an improvement.

Coding schemes

A crude printer cannot possibly print the full range of characters that TEX uses. So `Crudetype` tries to map each character onto the nearest equivalent in the printer's fonts, if any tolerable mapping exists. The mapping is defined in an array called *codes*. Since all characters on most crude printers are the same size, we need one piece of data, not for each TEX font, but for each coding scheme. For each character c in a TEX font whose coding scheme has internal number s, $code[s,c]$ defines the corresponding printer character. Also, $known\_schemes[s]$ is a character string which usually contains the coding scheme of that TEX font.

So when a font is read in, we try to determine which of the *known_schemes* it belongs to. If the printer is not absolutely crude, then it might have italic or bold fonts. Then we might want a coding scheme to correspond to a single TEX font. So first we look at the font name and see if that matches any of the *known_schemes*. But if the printer is fixed-width, then all fonts of the same face are the same size, so we drop the font size digits off the end of the name. If the font name is not in *known_schemes*, then we try again with the scheme given in the `TFM` file. If that fails, then the font is deemed to be unprintable and we do not load it.

Several crude printers (e.g. daisy-wheels) have only a limited set of characters, which cannot be extended. Sometimes you can generate more characters by overstriking. `Crudetype` can be programmed to do this, by placing suitable entries into a table called *ligatures*. The name is chosen by analogy with the *lig_kern* programs in `TFM` files, but the data is completely different. When one TEX character maps onto several printer characters, we call the image a 'multi-character' command.

Getting data into the *codes* array is clearly a very long and error-prone job, so special procedures were written to reduce this. First suppose that a run of consecutive characters in some TEX font maps onto consecutive characters in a printer font. The procedure *alphabet* will enter the whole run at one go. For example, to set up the AMTEX fonts (nearly ASCII) for a line printer, do:

```
known_schemes[1] ←
    'TEX␣EXTENDED␣ASCII␣␣␣␣␣␣␣␣␣␣␣␣';
alphabet(32,95,1,1,32);
```

The Standard requires that the coding scheme name be padded to the declared length. The parameters of *alphabet* are, in order, first character of TEX font, length of run, internal number of TEX font, printer font, and corresponding character of printer font.

Clearly, *alphabet* will only cover a very small part of the problem. The next procedure called *row* enters data into a subset of the *codes* array corresponding to a single row of a TEX font. In the standard font tables, row number m is the subrange $8m \,..\, 8m+7$ of a font. It is hoped that when the calls of this procedure are written out in a program, the result will be (just about) legible, whereas a string of statements like $codes[i,j].char \leftarrow 27$; is certainly not legible.

The main parameter of *row* is a character string that consists of 8 'character specifiers' separated by spaces. So a very simple call of *row* might be:

```
row('␣h␣i␣j␣k␣l␣m␣n␣o␣',2,13,1);
```

(As before, the string must be padded, but I have here removed the padding.) The numerical parameters are: TEX scheme, row, printer font. So this call of *row* will generate row 13 of TEX scheme 2, (TEX TEXT) which happens to be the same as the corresponding row of ASCII. In practice, we would never use *row* for such a simple purpose, because we would use *alphabet*.

There are several escape sequences that need to go into the row specifier string. Since all the PLAIN.TEX coding schemes (except the math extension one) have the upper case Roman characters in their ASCII positions, these characters will surely be inserted into *codes* by the *alphabet* procedure. So they are available as flag characters. But the brackets are also used as flags, as they are so much more intelligible than anything else. Some characters have most undesirable effects when used in `WEB` strings. So we make upper-case letters stand for them. 'A' generates an at-sign, 'Q' a single quote, and so on. 'L' says that the next character must be used literally. 'C' means that the next character must be replaced by the corresponding (ASCII) control character, and there are some further simple escape sequences.

Now things start to get rather complicated. *row* can also be made to generate multi-character commands, by bracketing several character specifiers together. Square brackets mean that the characters inside are to be overstruck, round brackets mean they are to be typed horizontally, and angle brackets mean that they are to be typed vertically above each other.

So to generate a Macsyma style summation sign, which looks something like this:

```
====
\
 >
/
====
```

we have to insert the following mess into the row specifier string:

```
<UUUU[====]\\[SL>]/[====]>
```

The 'UUUU' is needed to get correct vertical alignment. The 'L' is needed to prevent the following > being taken as an angle bracket. In order to keep track of what is happening and to provide some diagnostics, *row* has to impose some rather arbitrary rules of syntax. One of these is that character specifiers may not contain spaces. The 'S' is an escape for a space, and it is needed here to push the > one step right into its proper position.

Movements

This section considers the problem of deciding where each character has to be printed on the printer's page. This is by far and away the most difficult (and unsatisfactory) part of `Crudetype`. The current version is not a properly designed algorithm; it is merely a bodge, obtained by a lot of trial and error. It does seem to give tolerable results on `WEB` files, lineprinter, and VMS. We use these variables:

h is 'TEX's cursor'. It gives the 'exact' horizontal position (in `DVI` units) generated by `DVI` commands. This is always updated exactly as in `DVItype`.

hh is the 'printer's cursor'. It marks the position (in the printer's units) where the next character will be set.

The obvious method is to multiply *h* by a factor *h_conv* and round to nearest integer. This gives extremely bad results, because the characters in TEX fonts vary in width, while many crude printers have fixed-width characters. If *h_conv* is too large, then you get spaces in the middle of words. If *h_conv* is too small, then successive characters in a word get printed on top of each other. With an intermediate value of *h_conv*, you get both effects at once; in other words, the characters in TEX fonts vary so much in width that the 'too large' and 'too small' values of *h_conv* overlap. A great deal of jiggery-pokery is then needed to get a tolerable result (well, sometimes!). It is obvious that as soon as we begin to tamper with the exact rounding algorithm, *h* and *hh* will start to drift apart, so we must try to bring them together again. We want all the characters in each word to come together, and we want the accumulated drift to appear in spaces between the words.

So a second attempt to evaluate *hh* is as follows. On a crude printer, all simple characters have the same width (w, say), and usually $w = 1$. But multiple characters have different widths. So one of the things *row* must do when assembling a multi-character is to replace w by the correct value. Rules are in effect multiple characters. After we set each character, we increase *h* by the width and *hh* by w. Then we record the new value of *h* as *last_h*, and ignore all further changes in *h* until another character (or rule) is due to be set. (This 'lazy evaluation' on *hh* is not mere sloth but an essential part of the process). If $h - last\_h$ is small, we leave *hh* fixed. If $h - last\_h$ seems large enough to be a space between words, then we force *hh* to increase. If $h - last\_h$ is really large, we replace *hh* (provisionally) by:

$$new\_hh \leftarrow round(h * h\_conv);$$

This second attempt works a lot of the time on plain text, but often fails when TEX makes a large backspace. In fact TEX seems nearly always to do large backspaces by *pop* rather than an explicit move left. TEX often expresses boxes by a sequence like this:

```
PUSH Move right -----> [set characters] POP
  ↑                    ↑                 ↑
```

followed by a move either to one of the positions marked by the arrows, or close by. I try to deal with this by dropping markers at each of the arrowed positions. The right hand arrow is marked by *last_h*. The left hand arrow has just been popped off the stack; since the stack is realised as an array, it will be 'just above' the top of the stack. The centre arrow will be marked by *left_h*, which is defined as the value of *h* just before setting the first character after the latest *push*. Each marker has a corresponding value of *hh* attached. Suppose that we are about to set a character, and $h - last\_h$ is large and negative. Then we compare the current value of *h* with all the markers. Let m be the closest of these, and mm the corresponding rounded value. Then we re-round *new_hh* to force it to lie on the 'correct' side of mm. This seems to work fairly often, but it does sometimes slip.

Setting the vertical position on a crude printer is also very hairy. TEX expects subscripts to be much smaller than the main line, so it drops them by only a very small amount. On a crude printer, the subscript has to be the same size, and the

drop would normally get rounded to zero; it must be forced to be nonzero. When characters are underlined, TEX drops by a comparatively large amount, while the printer's underscore must be printed on the main line. So if v is the 'exact' vertical position and vv the rounded position, vv cannot be any monotonic function of v. What I have done is to declare a separate variable *rule_vv*, used only for vertical rules. As with horizontal moves, any large vertical move sets both vvs equal to the rounded value of v.

Sorting the page

Although 'crude' printers differ very much in their capacities, one thing they nearly all have in common is that they cannot feed the paper backwards. Some printers cannot backfeed at all; some tear the paper, and others let the paper slip and so lose position. Therefore it seems to be essential to process each page as follows: first copy the page into a suitable structure, then sort it by vertical and horizontal position, then print it.

The choice of method for sorting gave a lot of trouble. First I wrote the data onto a file and used the VMS library routines, but that had to be abandoned as not portable. Then I wrote a merge sort procedure which was amazingly slow. I believe this was because the files were being held on disc, and the disc transfers were slow. Shell sort was fast but not stable; I eventually chose a merge sort from 'Algorithms', by B.Sedgewick (Addison-Wesley, 1983). The algorithm is: chop the list in half, call *sort* recursively on each half, then merge the sorted halves.

The type of object that this algorithm sorts is a linked list. This could be represented either by a big array or by dynamic storage. Neither is ideal, because the size of a page is unknown, so whatever size you declare for an array is bound to be either too big or too small; and some PASCALs apparently do not implement pointers. So I have expressed everything in terms of certain macros, defined in the system dependent part of the program. Then `Crudetype` can be switched from heap to array merely by redefining these. For example, if p is logically a pointer, then the thing it points to will be defined as follows:

define *image*(#) ≡ #↑

when using dynamic store, and as:

define *image*(#) ≡ *pool*[*p*]

when using an array, here called *pool*. This illustrates one of the most valuable features of the `WEB` system: `WEB` not only makes it much easier to write programs, but it allows one to make complicated and far-reaching changes with much less difficulty than anybody could reasonably expect.

Known defects (July, 1986)

First it must be emphasised that this is an experimental version, offered 'as is' with no guarantee of performance. I do not have the time or manpower or machines to run adequate tests. Bug reports would be welcome, but the likeliest response will be something like "yes this is a bug and I do not know how to fix it; meanwhile, you have the source." Bug fixes are more welcome! That said, the principal known defects are:

1. Bad line breaks. When passed through `WEAVE` and TEX, `Crudetype` contains lots of these. It is of course perfectly possible to suppress bad breaks by inserting lots of *no_break* or *force_break* tokens into the `WEB` file, but I think it would be completely foreign to the spirit of TEX to do this. There ought to be a better way, and I hope somebody will tell me what it is.

2. Horizontal positioning. As explained above, this problem is very difficult, and I have only been able to produce a bodge that works sometimes. It is bound to fail sooner or later.

3. Downloading. `Crudetype` cannot support printers that can download characters. This is most unfortunate because it very severely limits the range of printers for which `Crudetype` is useful.

4. Accents. At least one make of printer does not provide accents, but accented letters. To print ü, you must send something like this:

```
<ESC>[2w     (that means Select German)
]            (an umlauted u)
<ESC>[10w    (Select ASCII).
```

It would be perfectly possible to make *row* generate this, but the real difficulty is that there is no obvious way for `Crudetype` to determine what character has been accented. A similar difficulty would arise with underlined characters, if one wanted to use `Crudetype` as a previewer (say, on a VT-100). An underlined character will generate character, backspace, underscore, and the underscore erases the character. Another problem that will make it difficult to use `Crudetype` to preview is that in order to conform to the Standard, it reads the `DVI` file sequentially. But any decent previewer must surely provide a Go to Page n command. So although the program makes some vague references to the alleged possibility that it might be used with a VDU, this is not yet really practical.

Site Reports

TEXhax returns

Editor's note: The following message arrived in electronic mail on September 23, 1986.

TEXhax "Neue Folge" Yr 86 Issue −1

This is a test run for the TEXhax distribution list.

TEXhax is about to get going again after more than a year of inactivity. I have worked through the very lengthy backlog of material and inclusion requests. The mailing list now reflects all changes and additions sent to `TeXhax-request` during that time. I suspect that a fair number of accounts that were on the original mailing listing are now no longer valid. Accordingly, I am sending around this dummy issue of TEXhax to 1) work out the kinks in the distribution table and 2) notify you of TEXhax' impending resurrection.

My name is Malcolm Brown. I am a consultant at Stanford and have been a member of TUG for several years. I'll be moderating TEXhax. Hopefully the first real issue will appear sometime next week.

Many thanks to Eric Berg at the Stanford Grad School of Business for assistance in working with the TOPS20 MM program!

Malcolm Brown

CDC Cyber Site Report

Jim Fox
University of Washington

The Office of Computing Services at the Georgia Institute of Technology has announced a port of TEX for the Cyber NOS/VE operating system. I have no details other than the person to contact. He is

Tharen Debold
Office of Computing Services
Georgia Institute of Technology
Atlanta, GA 30332-0275
(404) 894-4660
uucp: `tharen@gitpyr.uucp`
or `..!gatech!gitpyr!tharen`
Bitnet: `CC100TD at GITCDC1`

Porting TEX to the ATARI ST

Klaus Guntermann
Technische Hochschule Darmstadt

When ATARI came out with the 520 ST in 1985 we decided to buy some of them for use as a low cost raster graphics "terminal" for previewing TEX documents on its 640 × 400 dots black and white screen. Within 512 K bytes RAM memory which — at that time — was occupied by the operating system (about 200 K bytes) and the screen buffer (about 32 K bytes) it seemed to be impossible to run TEX. The 360 K bytes capacity floppy disks could not store such large programs either.

But soon there was a new version with 1024 K bytes main memory and double sided floppy disks with 720 K bytes capacity. That made a try to install TEX a challenging task. How fast would it be and could there be a floppy disk based TEX at all?

To complete our software toolbox we bought the ST development system including a C compiler and a lot of documentation. But we could not find a Pascal compiler for the ST powerful enough to compile TEX on the machine itself. Fortunately we had already a 68010 based TEX implementation on a multiuser system running MUNIX, a UNIX derivate operating system. The bugs of its Pascal compiler had already been circumvented by a lengthy change file. But there was no Pascal runtime system for the ST.

A close inspection of the MUNIX runtime system for Pascal showed that all necessary operations were done via the C runtime system and the system call interface. That suggested that we tried to combine the ST C runtime system and the MUNIX Pascal runtime system via some interface routines.

We sent the ST's C libraries via `kermit` to our MUNIX machine and developed two routines. One made the ST libraries look like a MUNIX library to the MUNIX loader, and a second converted the MUNIX relocation information into a format accepted by the ST loader. Again we were lucky that the subroutine calling sequence and the register usage of the ST libraries and the MUNIX compiler were compatible.

All the interfacing routines were written in C, because the MUNIX Pascal compiler allows C subroutine calls. The only part that had to be written in assembly language was the runtime startup that links the main program to the operating system, sets up the stack and the memory management.

In the last days of May, TEX (version 1.3) produced the first `DVI` output on the ST. A preview

installed on the MUNIX system and a printer driver for a dot matrix printer could be ported just the same way. The preview program required some additional changes because we wanted to make use of the ST's GEM interface, the pull down menus, file selector boxes and so on.

A "wizard" can use TEX and the preview program even on a single drive system tailoring the set of on-line fonts to the texts he wants to process and swapping disks occasionally. A dual drive system allows complete "TEXing". With the new packed font file format a reasonable set of fonts can be made available for previewing or printing in such an environment. The hard disk gives a speed up for loading TEX (about 9 seconds compared to about 40–60 seconds for TEX and FMT file) and faster access to fonts during preview or printer operation. Furthermore there is ample space for the complete font library in different magnifications for previewing as well as for printing on a dot matrix or laser printer.

Processing speed is comparable to the figures given by Tomas Rokicki in TUGboat Vol. 7, No. 2 for the Commodore AMIGA or the IBM AT if the hard disk is used.

For TEX 2.0 we added some new features. Memory is allocated dynamically for TEX such that there is no need to "unload" accessories that allow convenient access to useful features such as a desk top calculator or the standard VT52 terminal emulation (the memory occupied by them can only be freed by a system reset!), or to drop a RAM disk, just to process a small TEX file. If the minimum amount of memory is available (given by the memory size of INITEX) TEX can be loaded and started. But for large applications like LATEX it is wise to free about 730 K bytes to avoid a "sorry, TEX capacity exceeded" message.

The table sizes for font information, string indices and the strings themselves can be selected either at runtime or a customized TEX can be made by some simple patches to the object file.

In addition one may use the extended character set — with a lot of national characters — in a TEX input file. This makes TEX much more acceptable for a secretary who is used to type these characters if they are in the proper place on the keyboard, e.g. as it is in the German variant of the ST's keyboard. This is important especially in countries where accented characters or umlauts and the "sharp s" (ß) are very common in native language texts. The resulting loss of TEX source interchangeability can be overcome easily by a small program that makes the necessary simple replacements to get the standard 7 bit ASCII representation. This facilitates file transfer and processing by a "standard" TEX on some other machine.

Unfortunately the inclusion of national characters does not solve the hyphenation problem for words with accented characters/umlauts. But we hope that there will emerge a widely accepted standard for extension of the fonts by accented letters soon. The new METAFONT can generate them and TEX could access them via ligatures.

For more information about TEX, preview and available printer drivers for the ATARI ST, please contact

Kettler EDV-Consulting
P. O. Box 1345, D-8172 Lenggries
Federal Republic of Germany
phone (49) 8042 8081

The plan to use an ATARI ST as a "terminal" to preview documents without a local TEX has not been dropped. We hope to be able to report on such a distributed preview system in one of the next issues.

Editor's note: The TUG office has just been made aware of another implementation of TEX on the Atari ST, by

Tools, Ltd.
Kaiserstraße 48
5300 Bonn, Federal Republic of Germany

More information can be obtained from Edgar Fuß at that address. An article is promised for the next issue.

MVS TEX Site Report

Craig Platt
University of Manitoba

This is a short note to bring you up to date on my progress during the last year. I said in my last report (*TUGboat* Vol. 6, No. 3) that the MVS distribution tape should be available "Real Soon Now". Unfortunately I wasn't able to devote much attention to TEX during the academic year due to other duties, but a summer of feverish activity has yielded some progress.

At the TUG meeting at Tufts, I received Don's approval of my TRIP test results for TEX, version 2.0, as well as the TRAP results for METAFONT, version 1.0. (These tests had essentially been passed

much earlier, but this was the first time the results were presented for "official" approval.) I used the *ascii_tbl* file idea mentioned in my last report to allow creation of ready-to-run load modules for TEX and its friends. I also made a slight change in the file naming conventions. An explicit "area" prefix, as in `texinput:story.tex`, will now cause a search for members `story` or `storytex` in a partitioned data set with ddname `TEXINPUT`.

Until recently I was planning to distribute the tape through Maria Code, but there are a couple of minor problems. Maria Code has informed me that due to a change in duplicating service, they can no longer copy tapes at 6250bpi. All their tapes are 1600bpi on 1200 foot reels. I estimate that the material I have so far will take up about 1500 feet of tape at 1600bpi. It would be possible to ship 2400 foot tapes, but it would be more convenient to use the higher density. There could also be problems with users requiring special formats, so I am considering the possibility of distributing the tapes myself, at least for a while. This would make it easier to keep things up to date, especially while they are still being developed. The major question is whether I can spare the time. Watch this space for further news.

For the time being, I will send my current tape-in-progress for a handling fee of $50. This would be a standard label 1200 foot tape at 6250bpi containing a ready-to-run TEX and TEXware, including `plain` format. Sample JCL for loading the files is included, as well as a brief installation guide. I expect that a site would be able to get TEX up and running in a few minutes, after adjusting for the local character set.

I decided not to include AMS-TEX or LATEX format files (since I haven't installed them here yet), but a ready-to-run iniTEX is included so a site can generate its own. All change files for the above are included, as well as ready-to-run versions of `TANGLE` and `WEAVE`.

These are followed on the tape by the complete sources from the 2.0 (generic) TEX distribution from Stanford.

So—what am I waiting for? Here are a few things I hope to accomplish in the coming months.

- Ready-to-run METAFONT: I have included my change files for `METAFONT`, `GFtype`, `GFtoDVI`, and `GFtoPXL`, so that a `WEB`-fluent installer could compile them, but I would like to add ready-to-run versions as well, with sample JCL for generating the cmr fonts.
- Device drivers: The current tape contains no drivers, except for the `dviimp.web` source from the generic tape. I would like to include Pete Sih's 4250 and 3800 driver software from the CMS tape, if it can be made to run under MVS. (Roger Fajman at N.I.H. is working on this.) Bart Childs has offered to send me his generic driver from the Data General tape. Written in `WEB`, it might be easy to convert it to run under MVS. Finally, Nelson Beebe has a driver family written in C, which might be made to compile under either Waterloo C or Lattice C (from SAS). In any case, these drivers can serve as models for further development.
- Performance: Even though TEX and METAFONT pass the `TRIP-TRAP` tests, there is considerable room for improvement in speed. Chris Thompson at Cambridge sent me his change files, which contain several ideas I would like to incorporate, but haven't had time to test.

Finally, I'm a little reluctant to announce this as the "official" MVS tape because of my inexperience about distribution matters. In addition to the *ascii_tbl* and file naming conventions, I've made a number of arbitrary decisions regarding tape formats and blocking factors, based on not much more than guesswork. I would like to get a bit more feedback on the current version, and I welcome comments from all about the appropriateness of my choices.

For the benefit of others trying to compile TEX under `PASCAL/VS`, I should also mention a couple of bugs in release 2.2 of `PASCAL/VS`. When compiling TEX, the translation phase of the compiler produced the message:

```
AMPT99S TRANSLATOR ERROR
***IN STMT 29 OF ROUTINE CLOSEFILESAN
```

The traceback showed this was related to routine `MERGE` in module `AMPTOPT`. The fix was forwarded to me by Roger Fajman, along with that for another error he encountered, but which didn't affect my compilation. I will send these fixes to anyone who asks, and will include copies with the tapes.

To inquire about ordering, or about special tape formats, phone me at [204] 474-9832 during the day, or leave a message at 474-8703 and I will return your call. Please indicate the best time to call back. For network access, try

```
bitnet:  platt@cc.uofm.cdn
csnet:   platt%cc.uofm.cdn@ubc
arpa:    platt%cc.uofm.cdn%ubc.csnet
                 @csnet-relay
```

CMS TEX Site Report

Alan Spragens
Stanford Linear Accelerator Center

The CMS implementation of TEX version 2.0 was released shortly after the 1986 TUG meeting at the end of July. The tape includes TRIP- and TRAP-tested TEX 2.0 and METAFONT 1.0 in "load-and-go" form for the CMS environment, the Computer Modern source files, and the rest of the standard TEXware. Also included are Peter Sih's utilities to support the IBM 38xx and 4250 printers. Support tapes for these printers which will include pre-processed fonts are being prepared by volunteers at various sites and should be available soon. CMS TAPE DUMP format versions of the standard GF font tapes are also being made. CMS TEX has been improved, made smaller and faster but with greater capacity in this version, due to refinements contributed by several people, notably Chris Thompson of Cambridge University, U.K. The CMS METAFONT is due to Bernd Schulze of the University of Bonn.

The new tape incorporates a new ASCII-to-EBCDIC translation scheme adopted at the 1985 TUG meeting which differs from the previous CMS tapes in two characters: the ASCII caret is translated to hex 5F (EBCDIC logical not sign) and the ASCII vertical bar to hex 4F (EBCDIC solid vertical bar). Although this convention is expected to be more acceptable than the previous scheme for most sites, the ASCII-to-EBCDIC translation will continue to cause some consternation and will need to be worked out at each site. A small XEDIT macro, TEXCHARS XEDIT, is included on the tape to update TEX files prepared for CMS TEX 1.1 so they will work with the new system.

Another change appearing in the new system is in the format of TEX and METAFONT's non-text files such as DVI, GF, FMT, TFM, and BASE. All are now fixed record format with a record length of 1024 bytes. This should ease transfer of these files between various types of computers and allow programs which read and write them to handle the task in a more consistent and efficient way.

Editor's note: After several years of duty as the VM Site Coordinator, Alan is moving on. The new Site Coordinator is by no means a new face to the TEX community:

Dean Guenther
Computer Service Center
Washington State University
Computer Science Building, Room 2144
Pullman, WA 99164
509-335-0411
BITnet: Guenther@WSUVM1

Best of luck to both Alan and Dean.

Typesetting on Personal Computers

Real Typesetting from Your Personal Computer

Alan Hoenig and Mitch Pfeffer

TEX's ultimate goal is real typesetting. Now, 300 dpi laser output does look great, but Alfred A. Knopf would not find it acceptable for true camera-ready publication quality. You could have satisfied the late Mr. Knopf by exploiting the device independence of DVI files, and ship these to someone for output from a real typesetting machine. There are a handful of bureaus that perform this service for you, and the purpose of this column is to list them for you, with some facts and figures about each. We play no favorites; the organizations are listed below in strictly alphabetical order.

Please — if any readers know of organizations inadvertently omitted from this list, do get in touch with us; we'll run an update in a future column. If anybody knows of a printer manufacturer willing to support TEX output on their devices, please get in touch with either of us or with Barbara Beeton at the AMS.

Firms dealing with Autologic equipment can currently only deal with the AM (Almost Computer Modern) fonts. Autologic has been vague as to when they'll get their act together *vis à vis* the new CM fonts.

American Mathematical Society, P O Box 6248, Providence, RI 02940; (401) 272-9500

The AMS is happy to accept your DVI files on IBM PC or Macintosh diskette (although no PostScript extensions can be handled). Your output is produced for you by an Alphatype CRS, with a resolution of 5333 dpi. The turnaround is anywhere from one week to a month, with the average being two weeks. The AMS will be using the AM fonts for the next few months, but plans to change over to CM as soon as possible. The AMS is also looking to acquire a new typesetter in the near future, with access to the new machine's full font library.

The AMS has by far the most unusual price structure of anybody. There is a basic setup charge of $45. Next, you're charged 4.1¢ per square inch of copy. (For example, a sheet measuring $8\frac{1}{2} \times 11$ with 1-inch margins all around contains a body of type measuring $6\frac{1}{2} \times 9$ inches for an area of 58.5 in^2. You'd be billed $58.5 \times .041 = \$2.40$ for that page.) On top of that, you pay $2 for each sheet of film used to process the output. Sheets measure 16×20, and can easily accommodate four pages with a type area of $5 \times 8\frac{1}{2}$. Finally, there's a 12¢ per page charge for quality control.

Your contact at the AMS is Ron Whitney.

Ampersand Typographers, 176 Wicksteed Avenue, Toronto, Ontario, Canada; (416) 422-1444

Ampersand is the lone entry on this list who will definitely accept Macintosh disks, so people using FTL's *MacTEX* should take note. *MacTEX* includes a utility for transforming honest TEX DVI files to PostScript files, which are the objects Ampersand wants to see from you. Their output device is a Linotronic 300 with resolutions of 800, 1250, or 2500 dpi. (Ampersand recommends you stick with the medium resolution of 1250 dpi.) Charges begin at $10 per page and decline to $8 for more than 100 pages or so. There's an extra $7 per page charge for film output. Ampersand will accept any job no matter how small and guarantee a 24-hour turnaround, except for exceptionally large jobs. If you have additional questions, please contact Neville Robertson at this company.

Computer Composition Corp., 1401 West Girard, Madison Hts, MI 48071; (313) 545-4330

This company delivers your output to you within 48 hours. Your DVI files are reproduced on APS Micro-5 machines, and Computer Composition accepts IBM-PC diskettes and mag tape. As of this writing, C^3 has no facilities for accepting Macintosh disks, but if your job is large enough, they'll be glad to work something out with you. Charges range from $8 a page down to $4 for large quantities. Minimum job charge is $45.

C^3 can work with your source file to get your output in Times Roman and Helvetica, but that service might require an extra day. Be aware that page and linebreaks applicable to Computer Modern will be different for these non-TEX fonts.

For further information, contact Frank Frye, Computer Composition's president.

Stanford University is no longer accepting commisions from outside the university community.

TEXSource, 3333 W. Alabama, Suite 109, Houston, TX 77098; (713) 520-7206

TEXSource will accept your files on PC diskette, on mag tape, or by modem. (Sorry, no Macintosh.) Your material will be produced by an APS Micro-5, although soon they'll be acquiring an APS-6 which will be able to merge graphics with TEX text. Turnaround time is overnight except for horrendously large jobs, and the rates are $4 per page, dropping to $2.50 in large quantity. There's a $25 minimum charge.

I'll just mention some of TEXSource's other products of interest to TEX typographers. They can furnish a magtape unit for $3495 that hooks up to a PC or clone and reads data from a 9-track, 1600 bpi tape into the PC. Of great interest is their announcement that they'll soon be making Autologic fonts in 300dpi resolution available for use by TEX on PC's. Their first offering will include Times Roman and variations (italic, bold, and bold italic) in a variety of sizes for a price to be in the vicinity of $100. Intriguing.

Steve Bencze at TEXSource will be pleased to chat with you if you have further questions.

Textset, Inc., P O Box 7993, Ann Arbor, MI 48107; (313) 996-3566

In addition to developing various software-related products for the TEX environment, Textset will set your type for you on their APS-5. As usual, they'll accept your PC diskettes but not (yet) your Macintosh files. The turnaround varies between one day and one week. Charges range from $5.50 down to $4 per page, depending on volume.

Textset is willing to work to deliver your output using any of the several hundred typefaces in the Autologic library. Contact Gary Grosso at Textset for additional information.

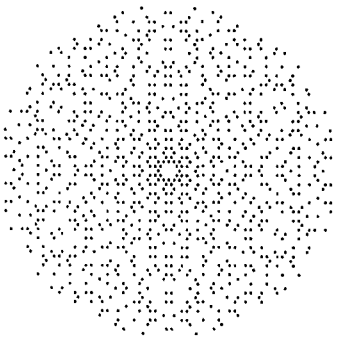
Universitätsdruckerei H. Stürtz AG Würzburg, Postbox 67 20, Beethovenstraße 5, 8700 Würzburg I, Federal Republic of Germany; telephone (09 31) 385-323

We have only limited information on this company, with whom we're familiar because of a mailing that doubtless went to many members of TUG. This firm appears ready, willing, and able to deliver TEX output on a Lasercomp with resolution of 1000dpi. Output in Monotype Times Roman is impressive and reportedly available in sizes from 4 to 90 points in 1/4-point increments. For additional information, contact Mr. Klingsporn or Mr. Tscheke at Stürtz. (We requested information ourselves, but we didn't get it in time to make the *TUGboat* deadline.)

Macros

Harnessing TEX to Compute Third Root of Unity Primes

Klaus Thull*
mahīlatā samāja



This image, showing the primes of $\mathbb{Z}(\sqrt{-3})$, up to norm 1000, was done by another "useless" macro set inspired by the `\primes` and the `\point` examples out of *The TEXbook*.

Mathematics: $\mathbb{Z}(\sqrt{-3})$ consists of all numbers of the form $(x + y\sqrt{-3})/2$ where x and y are rational integers, and $x + y$ is even. These numbers may also be expressed as $a + b(1 + \sqrt{-3})/2$ where a and b are rational integers. We need this later.

In $\mathbb{Z}(\sqrt{-3})$, there exists a Euclidean Algorithm. Thus, even as with the rational integers, there is a divisor theory, and prime numbers. The primes, instead of listing them as numbers, are mapped onto points of a Euclidean plane and plotted.

Any pair of numbers $q + r\sqrt{-3}$ and $q - r\sqrt{-3}$ is said to be conjugate. Their product is rational and is called their norm. The norm of a $\mathbb{Z}(\sqrt{-3})$ integer is a rational integer.

Two numbers of $\mathbb{Z}(\sqrt{-3})$ are associated if their quotient is a unit. The units are the powers of $(1+\sqrt{-3})/2$. There are six different units, arranged in a regular hexagon. Hence for each number there are six associates. The set of associates of a number may be equal to their conjugate, or disjunct. Yet in any case, their union obeys the D_6 (snow crystal) symmetry.

Each natural prime gives rise to a set of primes in $\mathbb{Z}(\sqrt{-3})$ which so obeys the D_6 symmetry. There are three cases:

- The *inert* case: The natural prime $2+3k$ is also prime in $\mathbb{Z}(\sqrt{-3})$. There is one self-conjugate set of six associates.
- The *split* case: The natural prime $1 + 3k$ is a product of two $\mathbb{Z}(\sqrt{-3})$ primes conjugate but not associate. There are two disjunct conjugate sets of six associates each.
- The *ramified* case: The natural prime 3 is an associate of a square of a $\mathbb{Z}(\sqrt{-3})$ prime. There is one self-conjugate set of six associates.

Macros: This describes how the image is done in TEX. The macros listed are clarified somewhat. All the `\new...` and the space gobblers are omitted as well as a few of the lowest level functions.

The outer main call: Note how the three different cases are handled.

```
\def\primes#1{\plot\@ne\@ne %  ramified
   \ri=7\rid=6\ru=#1 %            split
   \let\primeaction=\splitprimeaction
   \primesexecute
   \plot\tw@\z@ %                 inert
   \isqrt\ru\ru=\csqrt\ri=5
   \let\primeaction=\inertprimeaction
   \primesexecute}
```

Natural Primes: We compute the natural primes of the residue class given by `\ri` mod `\rid` in a range of numbers:

```
\def\primesexecute
   {\ifnum\ri>\ru\let\next=\relax
   \else\isitprime\ri\advance\ri by\rid
   \let\next=\primesexecute
   \fi\next}
```

Here is the primality test for one number. It is very simple: try division by 2, 3, and the residue classes 1 and 5 mod 6 until $\sqrt{\#1}$ or residue 0. Note that `\z@`, `\@ne`, `\tw@` etc. are TEX's constants for $0, 1, 2 \ldots$

```
\def\isitprime#1{\pc=#1\isqrt\pc
   \pl=\csqrt
   {\global\primetrue}
   \pt=\tw@ \tryprime \pt=\thr@@
   \tryprime \pt=\fiv@ \trynext
   \ifprime\primeaction\fi}

\def\trynext{\let\next=\relax
   \ifnum\pl<\pt\else
   \ifprime \let\next=\trynext
   \tryround\fi\fi\next}

\def\tryround
   {\tryprime \advance\pt by \tw@
   \tryprime \advance\pt by \f@ur}

\def\stateprimefalse
   {{\global\primefalse}}
```

* c/o Dittrich, Zeughofstraße 23,
D-1 Berlin 36, Germany

```
\def\tryprime{\rem\pc\pt
   \ifnum\wa=\z@\stateprimefalse\fi}

\def\splitprimeaction{\starter\pc}
\def\inertprimeaction{\plot\pc\z@}
```

Third Root of Unity Prime: For each split prime p in $\mathbf{Z}$, there is exactly one prime in $\mathbf{Z}(\sqrt{-3})$

$$a + b\frac{1+\sqrt{-3}}{2}$$

with a, b rational integer, $0 < b < a$ whose norm is p. To find this prime, we solve the diophantine equation

$$n = a^2 + ab + b^2 - p = 0.$$

The algorithm used here is $O(p^{1/2})$ yet cheap as such, and fast* for numbers $< 10^{15}$: Choose a fitting octant of the conic and start traveling at the concave side: $(a \leftarrow \lfloor\sqrt{p}\rfloor; \rangle b \leftarrow 0; \rangle n \leftarrow a^2 - p)$

```
\def\starter#1{\cn=#1
   \isqrt\cn\ca=\csqrt\cb=\z@
   \cn=\ca\multiply\cn by\ca
   \advance\cn by-\cx
   \onestep}

\def\onestep{\isit\ifnum\ca>\cb
   \oneadvance\let\next=\onestep
   \else\let\next=\relax\fi\next}

\def\oneadvance{\ifnum\cn<\z@\bp
   \else\bpam\fi}
```

weave out ... $(n \leftarrow n + a + 2b + 1; \rangle b \leftarrow b + 1)$

```
\def\bp{\advance\cn by\cb
   \advance\cb by \@ne \advance\cn by\cb
   \advance\cn by \ca}
```

and in: $(n \leftarrow n - a + b + 1; \rangle a \leftarrow a - 1; \rangle b \leftarrow b + 1)$

```
\def\bpam{\advance\cb by \@ne
   \advance\cn by\cb \advance\cn by -\ca
   \advance\ca by -\@ne}\relax
```

on solution ...

```
\def\isit{\ifnum\cn=\z@\ifnum\ca<\cb
   \else\action\fi\fi}
```

do something about it:

```
\def\action{\plot\ca\cb}
```

Plotting the primes: On obtaining one solution, we now compute all the associates and conjugates, thereby unskewing the coordinates such that the numbers in question are expressed as

$$\frac{x + y\sqrt{-3}}{2}$$

and \point the dots. This way, we retain integral coordinates. Yet, scaling y-unit by $1.7 \approx \sqrt{3}$ while \pointing yields a surprisingly close approximation to a true Euclidean mapping.

```
\def\plot#1#2{\ifnum#2=\z@\xa=#1
```

The inert case:

```
      \point{\xa}{\xa}\point{-\xa}{\xa}
      \point{\xa}{-\xa}\point{-\xa}{-\xa}
      \multiply\xa by \tw@
      \point{\xa}{\z@}\point{-\xa}{\z@}
   \else\ifnum#2=#1\xa=#1
```

The ramified case:

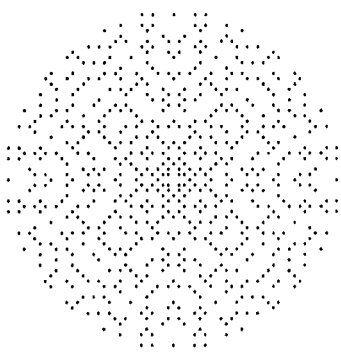
```
      \advance\xa by \xa
      \point{\z@}{\xa}\point{-\z@}{-\xa}
      \advance\xa by #1
      \point{\xa}{#1}\point{-\xa}{-#1}
      \point{-\xa}{#1}\point{\xa}{-#1}
   \else
```

The split case:

```
      \xa=#1\ya=#2\yb=\ya\advance\yb by-\xa
      \xb=\xa\advance\xb by \ya
      \xc=\xa\advance\xc by\xb
      \xd=\ya\advance\xd by\xb
      \point{\xc}{\ya}\point{-\xc}{-\ya}
      \point{-\xc}{\ya}\point{\xc}{-\ya}
      \point{\yb}{\xb}\point{-\yb}{-\xb}
      \point{-\yb}{\xb}\point{\yb}{-\xb}
      \point{\xd}{\xa}\point{-\xd}{-\xa}
      \point{-\xd}{\xa}\point{\xd}{-\xa}
      \fi\fi}
```

What else: Since no more $\mathbf{Z}(\sqrt{-3})$ mathematics are to be reported (and to keep this short) I left out the \point macro which follows the one recorded in the "Dirties" section except the y-unit scaling. Likeso the \rem macro, and the \isqrt (integral square root) which uses the Newton algorithm.

Exercise: Devise a set of TEX macros plotting this image:



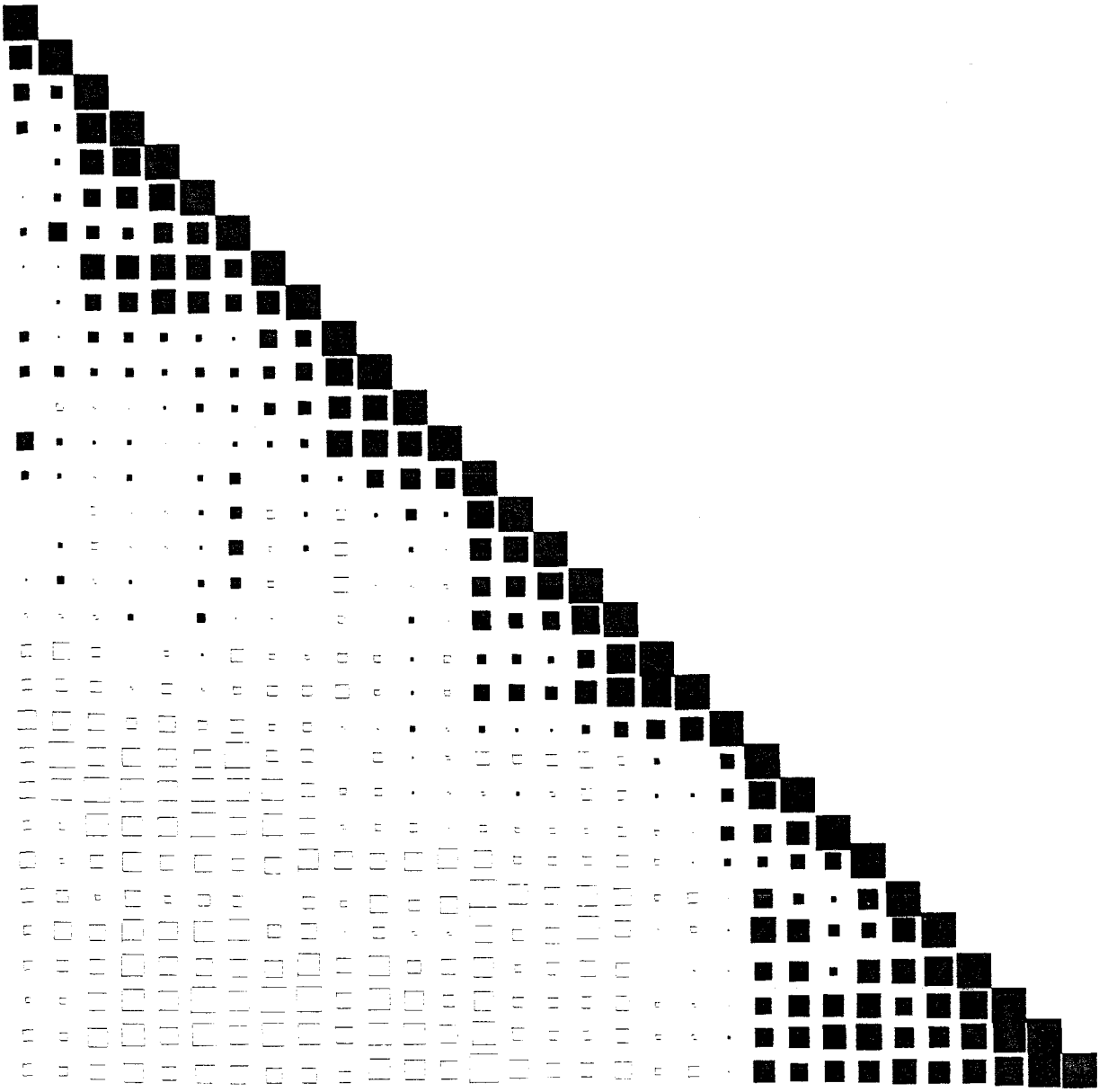
* I might have exited the algorithm upon the first solution found, speeding it up even more. This is left as an exercise.

Statistical Graphics With TEX

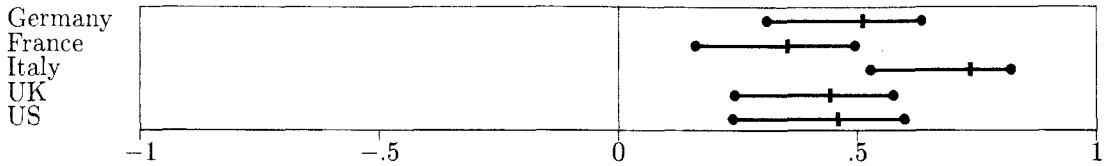
by Hans Ehrbar

TEX was not primarily designed for making graphical displays, and attempts to do graphics in TEX might be considered somewhat a stopgap measure. Still, it is tempting to experiment with this possibility, because of the convenience of putting the graphics directly into the text.

There are two other, less obvious, advantages to using TEX for graphics. First, the facility to make horizontal and vertical rules is well suited for certain kinds of graphics. The figure below is the printout of a correlation matrix (profit rates for two digit manufacturing industries 1949–79), in which the entries are replaced by squares, whose side lengths are proportional to the magnitudes of the entries. The solid squares represent positive, the bordered squares negative entries. This is a simple and attractive way of making the patterns in this correlation matrix visible.



The following figure, which shows confidence intervals, is another example for the use of rules for easy-to-read displays:



The small effort of having your statistics package (or computer program) give its output coded in TEX has also a second payoff. The high precision of TEX and its wide selection of fonts allow a lot of information to be compressed in a small space. Instead of reams and reams of computer printouts one can have all the information needed, together with some explanatory text, on a page or two. The scatter diagram on the following page of monthly data for capacity utilization (in percent, vertical axis) versus the inventory over sales ratio from February 1961 through November 1970 may serve as an example.

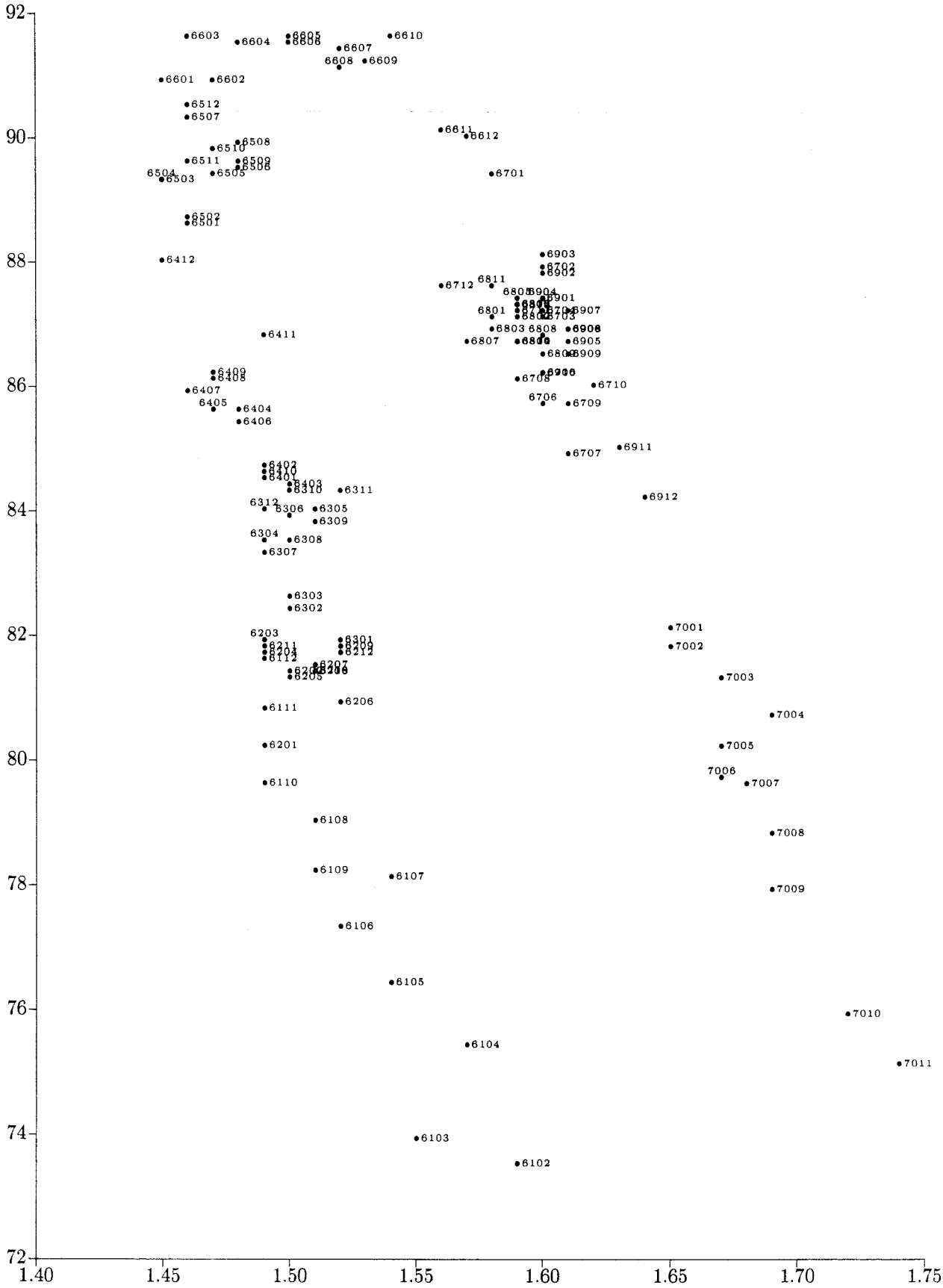
The syntax of the macros generating these displays is fairly simple, and should not provide any problems to someone familiar with the fundamentals of plain TEX. In fact, the macros are modeled according to math mode of plain TEX, i.e., they are variations of the alignment. The information necessary to produce these graphics (i.e., the coordinates of the scatter points, the endpoints and midpoints of the confidence intervals, the text to be written next to the scatter points, or the industry names in the confidence interval example) are arranged in a rectangular pattern, separated by & and \cr, which is given as an argument to a macro.

For the graphical representation of the correlation matrix, this is all one has to do. The effort is exactly the same as if one wanted to print the correlation coefficients in a matrix, but instead of \matrix one has to give the keyword \dotpattern, and instead of math mode one has to be in vertical mode (unless the display has only one line). Here is part of the actual code calling up the above dot-pattern:

```
This is a simple and attractive way of making the patterns
in this correlation matrix visible.

\medskip
\dotpattern{100
\cr 66.5992 &100
\cr 47.8082 &35.9022 &100
\cr 33.5024 &19.9712 &84.9131 &100
\cr 1.1944 &16.7153 &69.9001 &80.3869 &100

                                    ⋮

\cr -33.5713 &-22.9446 &-54.7992 &-63.8956 &-50.1286 &-65.7807 &-54.1421 &
-66.1329 &-61.4736 &-48.1490 &-64.2737 &-61.5697 &-52.0634 &-67.4709 &-35.6489 &
-27.6953 &-28.1442 &-40.6490 &-14.9838 &-13.9312 &8.3953 &54.1430 &60.5092 &
69.6542 &75.2284 &58.9060 &60.2270 &72.9116 &93.6946 &100
\cr -33.7705 &-26.5308 &-41.9840 &-54.9173 &-39.2742 &-53.4861 &-53.4312 &
-45.0544 &-44.0052 &-32.6934 &-59.9733 &-57.6638 &-52.3735 &-83.7067 &-54.0942 &
-41.3554 &-41.7640 &-49.4753 &-20.3350 &-31.3480 &4.1582 &63.3049 &52.5102 &
60.0084 &61.9984 &71.4205 &63.6169 &67.9257 &82.3474 &93.3305 &100
\cr  }
\bigskip

The following figure, which shows confidence intervals,
```

A little more has to be done for the graphs involving coordinate axes. The size of each graph is regulated by two dimensions, \hunit and \vunit, indicating the length of one unit on the horizontal and vertical axis. The scatter diagram needs a further detail: if the axes intersect at a different point than the origin, then one also has to assign to the dimensions \horigin and \vorigin values corresponding to the coordinates of the intersection of the axes.

The scatter diagram consists of three boxes joined together in horizontal mode: the first box contains the vertical axis, the second the scatter points (it has zero width), and the third box contains the horizontal


92
90
88
86
84
82
80
78
76
74
72
1.40
1.45
1.50
1.55
1.60
1.65
1.70
1.75
6603
6604
6605
6606
6610
6607
6608
6609
6601
6602
6512
6507
6611
6612
6508
6510
6511
6509
6506
6504
6503
6505
6701
6502
6501
6412
6903
6702
6902
6712
6811
6805
6904
6901
6801
6907
6703
6803
6808
6908
6411
6807
6905
6909
6409
6408
6708
6710
6407
6405
6706
6709
6404
6406
6707
6911
6402
6410
6401
6403
6310
6311
6912
6312
6306
6305
6309
6304
6308
6307
6303
6302
7001
6203
6301
6211
6209
6204
6212
7002
6112
6207
6205
7003
6206
6111
7004
6201
7005
7006
6110
7007
6108
7008
6109
6107
7009
6106
6105
7010
6104
7011
6103
6102

axis. The reference points of all three boxes is the "origin" of the diagram, where the two axes meet. Each of these boxes is produced by a macro, and all one has to do is to assign the appropriate values to \hunit and \vunit, and then call these three macros one after another in horizontal mode (e.g., inside a \centerline). Here one has to be careful that no blanks sneak in between these macros. The macros for the axes have as their first argument the distance between the ticks (measured in either \hunits or \vunits), and as the second argument an alignment with the text to be written next to (or below) the ticks. Here is the code invoking the scatter diagram:

```
\pageinsert
\vunit=1.1 cm \vorigin=72 \vunit
\hunit=45 cm \horigin=1.4\hunit
\centerline{%
\vertaxis{2}{92\cr90\cr88\cr86\cr84\cr82\cr80\cr78\cr76\cr74\cr72\cr}%
\scatter{73.50 &1.59 &6102
\cr 73.90 &1.55 &6103
\cr 75.40 &1.57 &6104

                                                    ⋮

\cr 79.70 &1.67 &&7006
\cr 79.60 &1.68 &7007
\cr 78.80 &1.69 &7008
\cr 77.90 &1.69 &7009
\cr 75.90 &1.72 &7010
\cr 75.10 &1.74 &7011}%
\horaxis{.05}{1.40&1.45&1.50&1.55&1.60&1.65&1.70&1.75}}
\endinsert
```

The scatter diagram has an additional feature: if one wants the text for a given point not next to the point, but above the point, one has to write it in the fourth column instead the third column of the alignment (as was done with the June 1970 scatter point).

If one does not want the scatter diagram but only the axes, one simply leaves the second macro out, or one can replace it with a different type of graphics.

The diagram with the confidence intervals consists of two boxes joined together in horizontal mode: the first contains everything except the ticks and measurements units at the bottom, and its reference point is the lower left corner of the rectangle framing the confidence intervals, and the second is built by the macro for the horizontal axis which we know from the scatter diagram. (The first macro can be considered to be replacing the vertical axis).

The macro building the first box has two alignments as arguments. The first argument contains the coordinates of the three vertical axes of the diagram (which need not be equally spaced), and the second argument contains the text and the coordinates of the endpoints and centerpoints of the confidence intervals (which need not be equally spaced, either). Again, the code should be largely self-explanatory:

```
\smallskip
\hunit=2.5 in
\centerline{%
\confival{-1&0&1}
{Germany&   0.309022&   0.511198&   0.634136 \cr
France  &   0.160441&   0.353459&   0.494577 \cr
Italy   &   0.526767&   0.737672&   0.822247 \cr
UK      &   0.242458&   0.443201&   0.575648 \cr
US      &   0.237897&   0.459010&   0.597618}%
\horaxis{.5}{-1&-.5&0&.5&1}}
\smallskip
```

The macros are so easy to use that your statistics package has to learn only one thing: compute the information to be represented in the graph in matrix form, and then print the matrix into the output file in the familiar pattern of an alignment.

The macros themselves are moderately complicated. The few tricks they use can be combined for also writing different graphical display macros. They are collected in the macro file **graphmac.tex** which is to be read in together with plain TEX. Arrangements will be made for TUG to distribute this file; the interested reader should call the TUG office for details.

Basically, these macros use (mis-use?) the alignment primitive for passing large and variable numbers of arguments to a macro. Often the entries in these alignments are not used for setting boxes, but only for assigning values to dimension registers. The **\dotpattern** macro could have been written in a way which follows more closely the spirit of an alignment; but in order to prevent box memory and macro memory overflow, the alignment only writes an auxiliary file **scratch.tex**, which is then read in to create the actual output. Besides writing the output file, **\halign** also creates a large number of empty boxes. In order to prevent these boxes from being appended to the main list (and thus wasting box memory), I used the **\setbox** command before invoking **\halign**, and afterwards emptied the box again.

When using these macros, it turned out that the size limitations of TEX (both macro memory when the alignment is read in, and then box memory when it is executed) are still rather stringent. I hope that larger versions of TEX will become available in the future—once everyone discovers how much can be done with TEX, constructions like the ones described in this note will no longer be the exception.

Economics Department
University of Utah
Salt Lake City, UT 84112

LaTeX

LaTeX — Call for Participation

Two volunteers have come forward to cover the LaTeX area for TUG. Jackie Damrau will be editing this TUGboat column, and Ken Yap will be collecting macros and style files in machine readable form.

For TUGboat

If any member has a specific question that he or she would like answered, or a macro to be published, please submit them to

Jackie Damrau
Editorial Assistant
Department of Mathematics
University of New Mexico
Albuquerque, New Mexico 87131

Creating a LaTeX Toolbox

TUG is actively soliciting your contributions for a machine readable collection of LaTeX style macros and related tools. This repository will be maintained at Rochester.ARPA. Leslie Lamport has kindly consented to review submissions.

Rochester.ARPA is accessible via anonymous FTP. Rochester is also on CSnet and Usenet. Bitnet access is via gateways. Tape and floppy distribution is a future goal. A table of current contents will be sent to appropriate mailing lists at regular intervals.

So, polish up that macro package or preprocessor and tell us about it. We are at:

`LaTeX-style@rochester.arpa`

Ken (Keeper of the Collection)
University of Rochester

Editor's note: Response to Ken's network message has been so prompt that he had the following to report before our printer's deadline.

Machine readable copies of user contributed LaTeX styles and `bib` style files are kept at `Rochester.Arpa`. To access this repository, FTP to `Rochester.Arpa` as anonymous with any password. Change directory to `public/latex-style` and retrieve the file `00index`, which gives a summary of what is available. I regret that I cannot satisfy requests to mail files due to lack of time.

Here are the contents, as of Sept. 24th 1986:

```
00index              siam.sty
00readme             siam10.doc
acm.bst              siam10.sty
doublespace.sty      siam11.sty
drafthead.sty        siam12.sty
ieeetr.bst           vdm.doc
siam.bst             vdm.sty
siam.doc
```

More submissions are welcome. Please mail your contributions to `LaTeX-Style@Rochester.Arpa`. Remember to include adequate documentation.

Ken, `LaTeX-Style@Rochester.Arpa`

LaTeX Bugs

Leslie Lamport

Two bugs have surfaced in LaTeX's figure-placement algorithm. They seem to be rare — one was reported to me by a single user, the other I discovered by myself — and are avoided by simply moving a figure or table environment in the input text. I think that one can be fixed and the other made even less likely to occur. However, this would require fiddling with some of the most delicate and obscure interaction between LaTeX and TeX, and I'm afraid that I'd be likely to introduce worse bugs.

LaTeX now seems to be fairly reliable, and I'd like to keep it that way. I am inclined to let these bugs stand, adding a warning to the Local Guide. However, before making such a decision, I'd like to find out if anyone else has encountered these bugs and how users feel about it. I can be reached as

`lamport@SRC.DEC.COM`

on the Arpanet.

Here are the bugs:

1. A figure or table may appear on the page preceding its appearance in the text. (Its position in the text will be at the top of the following page.) This is corrected by moving the environment one or two lines down in the input file. [This bug can't be completely fixed, but can perhaps be made less probable.]

2. A figure or table may be put on the same page as its appearance in the text when there isn't quite enough room, causing a footnote that should fit entirely on the page to be split across two pages. Moving the figure down a line or two in the text will correct the problem.

Chapter Mottos and Optional Semi-Parameters in General and for LaTeX

Reinhard Wonneberger
Hamburg*

Abstract

Motto texts will cause some logical and practical difficulties, when they are to be prefixed to chapter headings. To solve them, the motto text should be specified after the chapter heading. To allow for a variety of contained constructs such as footnotes or verbatims, this text must not be read as an ordinary parameter. These antagonistic goals are reconciled in a construct which looks like a parameter but is treated as input text. This concept is a modified version of the technique used for PLAIN footnotes. We give all macros necessary to implement this concept as an extension to LaTeX.

1 On Mottos

To provide the reader with a glimpse of what is waiting for him, a book or its chapters are sometimes prefixed with mottos. The basic idea of mottos, going right into the heart of a text in one short sentence, can be traced back to the times of ancient Babylonia, the myth of *Atramḫasīs* starting with such a motto verse:[1]

inūma ilū awīlum
When the gods were (also still) men ...

From a linguistic point of view, mottos are somewhat similar in function to particles, being both part of the text and a comment on it. So they are better understood in terms of a metatext.[2]

There is a wide range of possible motto texts, reaching from witty to ænigmatic, from aphoristic to devotional, from past to present. Normally motto texts will be quotations from some celebrity, but nowadays *graffitti* representing the *vox populi* will also be found.

As far as typesetting is concerned, graphic arrangement of mottos should meet several requirements. The special kind of text will be made clear through emphasis or even a different family of character type, e.g. *sans serif*. A scope, i.e. the range of text the motto applies to, will be expressed by prefixing the motto to an already established unit like a chapter. And finally, æsthetic concepts should be taken into account. So the motto will normally be broken into smaller lines, which may be right-adjusted to stress the frame of the page in connection with a heading. Professional book designers should be consulted on a specific concept for formatting.

Since mottos are normally taken from some source, one might also wish to put names into the index or give some bibliographic information in a footnote. Though there seems to be nothing peculiar about these requirements, implementation of mottos needs

* The macros presented here were developed at *DESY, Notkestraße 85, D 2000 Hamburg 54, FRG*. Comments should be sent to *R. W., Drachenstieg 5, D 2000 Hamburg 63* or through *Bitnet/Earn* to B03WBG at *DHHDESY3*.

1 Wolfram von Soden: Mottoverse zu Beginn babylonischer und antiker Epen, Mottosätze in der Bibel. In: W. v. S.: Bibel und Alter Orient. Altorientalische Beiträge zum Alten Testament. Hans-Peter Müller (ed.). *Berlin / New York: de Gruyter* 1985, p. 206 from p. 206-212.

2 It is fascinating to watch the gradual emergence of such metatexts, which are also used to give direct access to texts, a history still waiting to be written. For some remarks, cf. R. W.: Normaltext und Normalsynopse. Neue Wege bei der Darstellung alttestamentlicher Texte. Zeitschrift für Sprachwissenschaft 3 (1984) 203-233; cf. also R. W.: Leitfaden zur Biblia Hebraica Stuttgartensia. *Göttingen: Vandenhoeck & Ruprecht* 1984, chapters 3-5.

finding a solution that is upward-compatible. Our macros should also work for an alternative (asterisk) chapter command. A mottochapter macro looking quite similar to a chapter macro might look like this:

```
\mottochapter[toc_entry]{heading}%
{Here comes the motto text,
preferably from Shakespeare
\index{Shakespeare}
or the Bible,\footnote
{Use RSV \index{RSV}
to produce special effects!}
enclosed in braces.}
\par First paragraph of the chapter.
```

The only difference from the normal `\chapter` macro is the additional third parameter.

Since characters are assigned category codes when they are read from the file, and since category codes once set cannot be changed afterwards, our requirements cannot be met when the motto text has to be read as a parameter. What we would need then is the facility to specify the motto text as a parameter, but to read it as ordinary text. This sounds contradictory, but it is nevertheless possible in TEX, and to have a name for this construct, we shall call it a *semi-parameter.*

3 Semi-Parameter Footnotes

The concept of semi-parameters has already been realized in the footnote macros described in *The TEXbook* p. 363, whereas the restrictions we mentioned above also apply to ordinary LATEX footnotes, which read the footnote text as a parameter. To avoid these restrictions, we modified the LATEX footnote macros in this respect. Since it requires some care to adapt the general technique to LATEX control sequences, it might be helpful to give the actual code here.[8] An explanation of the basic technique will be given in the next section.

[8]

```
%\head footnotes
% The following macros avoid to read the footnote
% text as parameter.  They are taken from PLAIN.
% LaTeX control sequences have been substituted
% for those from PLAIN, marked by              %LATEX.

%   \footnotesep     : The height of a strut placed
%                      at the beginning of
    \footnotesep=0pt
%           do not sep footnotes for esthetic reasons
```

Though the technique is basically the same as the one explained before, the semi-parameter here

```
% LaTeX keeps PLAIN TeX's \footnoterule as the default
%\def\footnoterule{\kern-3\p@
%  \hrule width 2truein \kern 2.6\p@}
%         raise fnrule for esthetic reasons
 \def\footnoterule{\kern-3.9\p@
   \hrule width 2truein \kern 3.5\p@}

% \newinsert\footins
%\def\footnote#1{\let\@sf\empty
% \ifhmode\edef\@sf{\spacefactor\the\spacefactor}\/\fi
% #1\@sf\vfootnote{#1}}
% \def\vfootnote#1{\insert\footins\bgroup
\def\@footnotetext{\insert\footins\bgroup
  \footnotesize                                    %LATEX
  \interlinepenalty\interfootnotelinepenalty
% \splittopskip\ht\strutbox % top baseline for broken f.
    \splittopskip\footnotesep                      %LATEX
  \splitmaxdepth\dp\strutbox \floatingpenalty\@MM
% \leftskip\z@skip \rightskip\z@skip \spaceskip\z@skip
%           \xspaceskip\z@skip
    \hsize\columnwidth \@parboxrestore              %LATEX
\edef\@currentlabel{\csname p@footnote\endcsname %LATEX
\@thefnmark}%                                      %LATEX
% \textindent{#1} %Knuth
% \textindent{\@thefnmark} % yields normal numbers
\@makefnlabel
\footstrut\futurelet\next\fo@t}

% the following is an addition to     REPORT.sty
% format the footnote label inside the footnote
\def\@makefnlabel{\@@par \parindent 1em\noindent
\hbox{\lower0.25ex\hbox{\quad$^{\@thefnmark}$\quad}}}

% \long\def\@footnotetext#1{\insert\footins{\footnotesize
%    \interlinepenalty\interfootnotelinepenalty
%    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
%    \hsize\columnwidth \@parboxrestore
% \edef\@currentlabel{\csname p@footnote\endcsname
% \@thefnmark}
% \@makefntext
%    {\rule{\z@}{\footnotesep}\ignorespaces
%      #1\strut}}}

\def\fo@t{\ifcat\bgroup\noexpand\next \let\next\f@@t
  \else\let\next\f@t\fi \next}
\def\f@@t{\bgroup
          \rule{\z@}{\footnotesep}\ignorespaces  %LATEX
          \aftergroup\@foot\let\next}
\def\f@t#1{#1\@foot}
% \def\@foot{\strut\egroup}
\def\@foot{\strut\egroup
           \ifhmode\spacefactor\@x@sf\relax\fi}  %LATEX

\def\footstrut{\vbox to\splittopskip{}}
\skip\footins=\bigskipamount % space added
                             % when footnote is present
\count\footins=1000 % footnote magnification factor
\dimen\footins=8in % maximum footnotes per page

\def\f.{\footnote\bgroup} \def\ef.{\egroup} % local defs
```

is *obligatory*, and the next token will be used if no proper parameter is found.

The use of semi-parameters is a first step towards the solution of our motto problem. If we can also make this parameter optional, we can even achieve compatibility with earlier texts and save the effort to type empty motto parameters if no motto is present. This, too, is possible in TEX, and so we arrive at the concept of *optional semi-parameters*.

It should be noted, however, that this concept is not in accordance with the rules of LATEX, which require optional parameters to be enclosed in brackets ([...]). But this means parameter reading, just the thing we want to avoid. On the other hand, optional and also obligatory semi-parameters do not interfere with LATEX syntax, and in some cases are even an enhancement.

4 Optional Semi-Parameters

As we said before, the technique of semi-parameters is shown in the footnote handling macros of *The TEXbook* p. 363, but it is not explained there. Because this technique is of general importance, we should drag it out of the discouraging bunch of footnote submacros, and present it here with a few explanations. Our version will be different from the original one in two respects. First, to avoid conflicts with possible footnotes, we use the same code with different names; second, we modify the macros so that they will test whether the semi-parameter is present or not, thus allowing it to be optional instead of required as with footnotes.

```
\def\@readmotto{\begingroup \mottoformat
            \futurelet\next\mo@t} % 1

\def\mo@t{\ifcat\bgroup\noexpand\next
            \let\next\m@@t
            \else\let\next\m@t\fi
            \next} % 2

\def\m@@t{\bgroup\aftergroup\@motto
            \let\next} % 3

\def\m@t{\nomottoformat \endgroup
            \@aftermotto} % 4

\def\@motto{\endmottoformat \endgroup
            \@aftermotto} % 5

\def\@aftermotto{\%
      @printmottoheading{\@headingtext}
      \@afterheading
      \if@twocolumn \@endtopnewpage \fi} % 6
```

1. The first macro, after starting a group and the formatting environment for mottos, does nothing but load the next element into `\next` for inspection by the second macro. This element is not removed from the input. To make sure the outside world is not affected by our operations, we enclose everything to follow in a 'hard' group using the `\begingroup` primitive, which must be closed by an `\endgroup` primitive and so might help to detect grouping errors in the motto text.

2. The second macro will decide between a group (*if* case) and any other context (*else* case), choosing the latter also if the group follows after a space. This allows us to follow a mottoless `\chapter` macro with a normal group.

3. The third macro (*if* case) first opens a group to replace the group opening symbol that has been found by `\futurelet` before and will be swallowed at the end of this macro by `\let` (see *The TEXbook* p. 376). This is necessary because the `\aftergroup` action must be defined from *within* the group after which it is to be performed. Then it stores the macro to be executed after the group. This group will be the motto text semi-parameter read from input, and accordingly it will be closed by the group closing symbol of the motto text coming from the input. Finally it swallows the group opening symbol from the input text, in order to compensate for the group opening symbol it put in before.

4. The fourth macro (*else* case) is executed when no group follows immediately. It contains a macro that will typeset something other than the missing motto, typically some sinkage (vertical space). It closes the 'hard' group and executes the aftermotto macro.

5. The fifth macro supplies the corresponding end command for formatting, plus a group clos-

ing symbol for the 'hard' group opened in the first macro, and then executes the aftermotto actions.

6. The last macro contains the actions to be taken after the motto. If we are in `twocolumn` style, we should close here the corresponding page-wide heading box.

5 Formatting the Motto

These macros will execute a formatting macro at the beginning and at the end of the motto. *Leslie Lamport*, who read a draft of this article, strongly recommends that a professional book designer be consulted on the actual formatting of the motto. So, the following formatting is only meant as a default for draft purposes, and the corresponding macros have been given accessible names so that they can be easily redefined by the user.

```
% motto formatting example
\def\mottoformat{\hfill
%              position motto to the right
      \begingroup
      \minipage{0.63\textwidth}
%                       narrow motto column
      \parindent 0pt
%                  do not indent paragraphs
      \begingroup
      \flushright       % rightadjust text
      \mottofont}
%               use special font, e.g. \sf

\long\def\endmottoformat{%
      \endflushright
%               end rightadjustment of text
      \endgroup
%     \nobreak % penalty against page break
      \endminipage
      \endgroup
      \par}

\def\nomottoformat{%
      \vspace*{10ex} % produce sinkage here
      \relax }

\let\mottofont= \em
%                   default for motto font
```

The formatting of the motto text is defined here using only constructs of standard LaTeX. It can also be used as a separate environment to format a motto without connection to a chapter, which can be useful to test the best shaping of motto texts. To allow this, the normal form of LaTeX environments, `\begin{flushright} ... \end{flushright}`, should be replaced by the internal form enclosed in a hard group to preserve the original structuring, e.g. `\begingroup \flushright ... \endflushright \endgroup`.

When footnotes are used inside the motto, they will be placed into the box made by the `minipage` environment. This `minipage`-default was chosen because it will work also in the case of `twocolumn` style, where footnotes would otherwise just disappear. Even then, the technique of splitting a footnote into a 'callout' and the actual note can be used to obtain ordinary bottom footnotes. In many cases it will be best to specify linebreaks explicitly inside the motto with \\ (newline) commands. Then it will be appropriate to leave out the minipage environment. Examples for both cases will be given in the appendix.

There is also a `\nomottoformat` macro, which allows us to specify appropriate actions if no motto is present. The default will be to add some vertical space (*sinkage*) instead of the motto.

6 Upgrading the LaTeX chapter command

If the chapter macros of LaTeX are studied carefully, it turns out that motto printing has to be done after the test for `twocolumn` style has been performed in the `\@chapter` and `\@schapter` macros, and before the group for printing the heading is started in the `\@makechapterhead` and `\@makeschapterhead` macros. In fact, the motto production should replace the command immediately before this group, which is a `\vspace*{...pt}` control sequence. This command produces the so-called *sinkage*. There is also a close graphic connection between motto and sinkage: if a motto is present and printed on the right part of the page, a natural sinkage is produced, and the vertical distance between chapter heading and motto can be much smaller than otherwise.

However, the problem cannot be solved just by changing this particular spot. The remainder of

the macro has to be removed and associated with the macro that will end a motto. Things are even more complicated for `twocolumn` style, which uses a `\@topnewpage[...]` macro that introduces another level of nesting. Though our postponing technique might be used again, it seems easier to split the `\@topnewpage[...]` in a beginning and end part, and integrate it into the `\@makechapterhead` and `\@makeschapterhead` macros.[9]

We also have to make sure that our inspecting macro `\@readmotto` will be the last thing to be executed in the macro it is called from. Thus we have to modify the `\if ... \fi` construction at the end of the `\chapter` macro to avoid calling it from within such a construction.[10] Another important technique that could be used is to assign the next action to be taken to an intermediate control sequence (`\next`) as explained in *The TEXbook* p. 352.

To use the new technique, we can include the new macros and `\let \chapter= \mottochapter`. Previous texts can be run without disturbance, and even when run without our macros, the new input will do no harm: the motto text will just print as normal text after the chapter heading.

7 Conclusion

Our discussion might have a whiff of "Much Ado about Nothing", were it not for the general impor-

9

```
% for normal section %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\@makechapterhead#1{\def\@headingtext{#1}\let
      \@printmottoheading=\@makemottochapterhead
   %  \tracingall
      \if@twocolumn \@begintopnewpage \fi
      \@readmotto}

\def\@makemottochapterhead#1{\vspace*{3ex}
  { \parindent 0pt \raggedright
    \ifnum \c@secnumdepth >\m@ne
%                   IF secnumdepth > -1 THEN
      \huge\bf
      \@chapapp{} \thechapter
%                   Print 'Chapter' and number.
      \par
      \vskip 20pt \fi
%                   Space between number and title.
    \Huge \bf          % Title.
    #1\par
    \nobreak      % TeX penalty to prevent page break.
    \vskip 40pt } }   % Space between title and text.

% for star-section %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\@makeschapterhead#1{\def\@headingtext{#1}\let
      \@printmottoheading=\@makesmottochapterhead
      \if@twocolumn \@begintopnewpage \fi
      \@readmotto}

\def\@makesmottochapterhead#1{\vspace*{3ex}
  { \parindent 0pt \raggedright
    \Huge \bf          % Title.
    #1\par
    \nobreak       % TeX penalty to prevent page break.
    \vskip 40pt } }    % Space between title and text.
```

10

```
\def\chapter{\clearpage  % Starts new page.
   \thispagestyle{plain}
%                Page style of chapter page is 'plain'
   \global\@topnum\z@    % No figures at top of page.
   \@afterindentfalse
%             No indent in first paragraph, otherwise
%             change to \@afterindenttrue
   \secdef\@chapter\@schapter}

\def\@chapter[#1]#2{\ifnum \c@secnumdepth >\m@ne
        \refstepcounter{chapter}
        \typeout{\@chapapp\space\thechapter.}
        \addcontentsline{toc}{chapter}{\protect
        \numberline{\thechapter}#1}\else
      \addcontentsline{toc}{chapter}{#1}\fi
   \chaptermark{#1}
% Add between-chapter space to lists of figs & tables:
   \addtocontents{lof}{\protect\addvspace{10pt}}
   \addtocontents{lot}{\protect\addvspace{10pt}}

 %begin of modifications: %%%%%%%%%%%%%%%%%%%%%%%%%%
  \@makechapterhead{#2}}
 % will replace:
 %   \if@twocolumn % Tests for two-column mode.
 %         \@topnewpage[\@makechapterhead{#2}]
 %   \else \@makechapterhead{#2}
 %         \@afterheading  % Routine called after
 %   \fi}               % chapter and section heading.

\def\@schapter#1{\@makeschapterhead{#1}}

% split \long\def\@topnewpage[#1]{...} in two parts: %%%
 \def\@begintopnewpage{\@next\@currbox\@freelist{}{}
     \global\setbox\@currbox\vbox\bgroup
     \hsize\textwidth \@parboxrestore}

 \long\def\@endtopnewpage{\par
     \vskip -\dbltextfloatsep\egroup
     \global\count\@currbox\tw@
     \global\@dbltopnum\@ne
     \global\@dbltoproom\maxdimen\@addtodblcol
     \global\vsize\@colht \global\@colroom\@colht}
```

tance of the techniques that it leads to. First of all, there is the technique of *semi-parameters*, which we just borrowed from *The TEXbook*.

Semi-parameters will allow us to perform some actions *after* they have been processed, resembling normal parameters in this respect. But as they are not read like normal parameters, they do not impose restrictions on the type of constructions that they may contain. Thus they allow nesting of "dangerous" operations; for example, the index macro for our book *Verheißung und Versprechen*, which requires special treatment of category codes, can be part of a footnote which is in turn part of a motto. On the other hand, obligatory semi-parameters do not interfere with LaTeX syntax. As is shown by the `\footnote` case, they are an enhancement and upward-compatible at the same time.

Second, we found a way to make this type of parameter *optional*. As we noted before, this concept is not in accordance with the rules of LaTeX, which have either obligatory normal parameters or optional parameters enclosed in brackets (`[...]`). But in both cases parameter reading is implied, just the thing we want to avoid. Though not much harm is done, texts with optional semi-parameters are no longer compatible with standard LaTeX, unless the corresponding macros are also included. But since they might prove useful in other cases too, and existing LaTeX input is not affected, it might be worthwile to consider them for the next major release. And I should like to add that in my opinion LaTeX is too important a tool to be frozen already in its present state.

In coming to the end of this article, we should give some thought to its history. When it became clear to me that the motto problem was not trivial, I did not start by writing macros to solve the problem, but by writing explanations for macros that did not exist at that time. The macros were written only after things had become clear to me through the process of compiling the explanations. I am quite convinced that this saved me more testing time than it cost me in writing, while giving other people access to the concepts of these macros at no additional cost and thus saving them the burden of exploring ways that lead to nowhere.

This approach was supported by the file inclusion mechanism of our operating environment, which allows us to include a macro everywhere. So we can use the same source code to be executed and to be printed as a verbatim listing. As far as I can see, this is not possible in LaTeX, but might be implemented using the technique described in *The TEXbook* p. 380f.

Modifying existing macros and writing new ones could be improved a great deal if, instead of some sparse comments in the source files, we had a WEB-like style of macro development and description, and this article should be seen as a first attempt to move towards a technique that has proven to be one of the most successful tools in software engineering.

On aurait souhaité de n'être pas technique.
A l'essai, il est apparu que,
si l'on voulait épargner au lecteur les détails précis,
il ne restait que des généralités vagues,
et que toute démonstration manquait.
ANTOINE MEILLET,
Esquisse d'une histoire
de la langue latine, 1928.[11]

Appendix A

Testing Mottos

Our macros are based on the `\chapter` command, which is not present in the article style used here. In order to test them, we first have to include the macros for chapters from the `rep12.sty` or a similar file and then our own, so that a complete set of macros will emerge. Then we `\let \mottochapter= \chapter` and give our text, which is shown in the footnote to our example motto.

Note that we used motto formatting without a minipage environment, controlling the linebreaks explicitly according to the parts of the sentence.[12]

[11] Quoted from CURTIUS LITERATUR 7. — Note that even verbatim text can be contained in a footnote being part of a motto:

```
\chapter{Testing Mottos}%
{On aurait souhait\e. de n'\ec.tre pas technique.\\
A l'essai, il est apparu que, \\
si l'on voulait \e.pargner au lecteur les d\e.tails pr\e.cis,\\
il ne restait que des g\e.n\e.ralit\e.s vagues,\\
et que toute d\e.monstration manquait.\\
{\em {\sc Antoine Meillet,}\\
Esquisse d'une histoire\\ de la langue latine, 1928.\footnote
{Quoted from {\sc Curtius Literatur} 7.%
}}}% end of footnote, end of \em, end of motto
```

[12] This principle was used to typeset a whole book on books for children; an example is reproduced in my article *Normaltext und Normalsynopse* (see above) in Fig. 20 on p. 225.

To close our article, we use a `mottoformat` environment containing a `minipage` environment[13] to highlight the words found at the end of the account of *Sinuhe*:

> *It has come (to its end)*
> *from beginning to end*
> *as it had been found in writing.*[a]
>
> [a] James B. Pritchard: Ancient Near Eastern Texts Relating to the Old Testament. Princeton 1969, p. 23.

[13]

```
\begin{quote}
\begin{mottoformat}
It has come (to its end)\\
from beginning to end\\
as it had been found in writing.\footnote
{James B.~Pritchard:
Ancient Near Eastern Texts
Relating to the Old Testament.
Princeton 1969, p.\,23.}
\end{mottoformat}
\end{quote}
```

A LaTeX Addition for Formatting Indexes

Thomas Hofmann
Ciba-Geigy, Basle, Switzerland

A query by Jim Ludden (TUGboat, Vol. 7, No. 2, page 111) asked for a LaTeX addition for formatting indexes. Here is my proposal for UNIX systems: `latexindex`, a Bourne Shell script based on `texindex` by Robert Plamondon (location unknown). When using `latexindex`, there is no manual work on the `.idx` file.

`Latexindex` needs the four files `index.awk`, `index1.awk`, `index.sed`, and `index1.sed` whose full pathnames have to be set in the variables `INDEXAWK`, `INDEXAWK1`, `INDEXSED`, and `INDEXSED1` at the top of the file `latexindex`.

Description of `latexindex`

`Latexindex` processes an `.idx` file created by LaTeX and generates a `theindex` environment. The extension `.idx` for the input file name may be omitted. The result is left on the standard output. Example: If `test.tex` includes a command `\input test.ind`, call `latexindex test >test.ind` after having LaTeXed test, and then start LaTeX again.

The page numbers for entries of the form `\index{`⟨*entry*⟩`}` appear in roman font, for entries of the form `\index{`⟨*entry*⟩`*}`, in bold-face. The asterisk only marks a bold-faced page number at the end of an entry, otherwise it is regarded as part of the entry.

If an index entry appears on the same page both with and without an asterisk, the entry without the asterisk will be ignored. Consecutive page numbers are turned into ranges; however, bold-faced page numbers always appear separately. (Example: Index, 2–5, **6**, **7**, 10–11)

For sorting there is no difference between uppercase and lowercase letters. Digits fall before the letter 'A'. Special characters are ignored. However, two entries which differ only in special characters are not combined. Therefore `\index{Index}` and `\index{In\-dex}` yield two different entries.

Subentries in index commands must have the form `\index{`⟨*main entry*⟩`,`⟨*subentry*⟩`}`. A subentry appears indented under the main entry. If a main entry has several subentries, the main entry appears only once, followed by a sorted list of subentries. It is not possible to create subsubentries. An entry without subentry may not include commas.

An index command having the form `\index{[`⟨*entry1*⟩`]`⟨*entry2*⟩`}` yields an index entry ⟨*entry1*⟩ but sorted by ⟨*entry2*⟩. In this way, font switches and other LaTeX commands can be included in the index without using the command names for sorting. Example: `\index{[\it Index]Index}` allows the italic index entry *Index* to be sorted by "In", not by "it". When using subentries, both the main entry and the subentry have to appear twice (e.g. `\index{[Document style,\tt article]Document style,article*}`). For a bold-faced page number in this case the asterisk has to be placed, in the same way as described above, immediately before the closing curly brace (and not within the square brackets). Commands changing the environment which appear within brackets or curly braces lose their effect outside. A command within the main entry that changes the environment is not effective for the subentry (for `\index{[\it Index,Fonts]Index,Fonts}` only "Index" appears in italic, "Fonts" appears in Roman).

Bugs

- An opening curly brace must not immediately follow a closing one "}{".
- Neither the symbol '`@`' nor brackets may appear in index entries, except to mark the sorting entry. If these symbols have to appear in the index, they must be defined by appropriate commands. (Example: Instead of `\index{Program[me]}` you must type
  `\newcommand{\lbr}{[}`
  `\newcommand{\rbr}{]}`
  `\index{Program\lbr me\rbr}`.)
- There may be only *one* comma within a sorting entry (for separating the subentry), and there may also be only one comma in the brackets.
- Main and subentries must not be nested in common curly braces (except the outer braces of the `\index` command). E.g. the command `\index{{\it Index,Bugs}}` is not allowed; the correct form is
  `\index{{\it Index},{\it Bugs}}`, or simply
  `\index{{\it Index,\it Bugs}}`,

Editor's note: Arrangements are being made to include the files comprising `latexindex` in the Unix distribution, and perhaps to make them available separately through TUG.

IdxTEX and GloTEX Indexes and Glossaries

Richard L. Aurbach
Monsanto Company

With reference to Jim Ludden's request for a program to format LATEX indices, we have developed two programs here at Monsanto which you may find interesting.

IdxTEX

The IdxTEX program is an automatic index generator for LATEX. It features three-level indexing, visual highlighting of index entries and page number references, and generates a file which may be `\input` into your document to produce a fully-automated and fully-formatted index.

GloTEX

The GloTEX program is to glossaries what BIBTEX is to bibliographies — it is a program which uses databases of definitions to produce nicely-formatted glossaries, using the `\makeglossary` and `\glossary` features of standard LATEX.

Both of these programs are being made available to TUG for distribution to interested parties. The programs are VAX/VMS specific, but are written in C and are (I hope) exhaustively documented. They are not portable (because they call VAX/VMS services), but should be relatively easy to port to other environments. The distribution kit contains executable images, full sources, users' guides (and the special document styled needed to generate them), and on-line help modules.

I request that anyone who enhances these programs, finds and fixes bugs, or makes any other changes in these programs (other than porting them to other environments) let me know so we can also benefit. I would also be interested in ports to the VM/CMS environment.

Queries

Editor's note: When answering a query, please send a copy of your answer to the TUGboat editor as well as to the author of the query. All answers will be published in the next issue following their receipt.

In addition to the item below on change bars (query by Sylvester Fernandez, Vol. 7, No. 2, page 110), two responses to the query by Jim Ludden (*ibid.*, page 111) regarding post-LATEX index formatting appear in the LATEX column, beginning on page 186.

Form Letters

Is there a package available that allows the generation of identical letters *except* for the addressee field, which is read from a separate file containing lines of address separated by a delimiter?

John Lee
jslee@nrtc.arpa

Change Bars

Jim Fox
University of Washington

The question of change bars was asked in the last TUGboat. Here is how we do them at the Academic Computing Center.

start change bar:

```
\special{changebar, \the\barwidth}
```

end change bar:

```
\special{changebar}
```

where `\barwidth` is a ⟨dimen⟩ register that describes the width of the bar. A vertical rule will be drawn from the location of the first special to the vertical coordinate of the second special. Note that the second special only defines the vertical extent of the rule — its horizontal coordinate is ignored.

Macros that are compatible with `plain.tex` output routines do the positioning automatically —

the user need only type \beginbar and \endbar. Marks are used to stop and start multiple page change bars.

The change bar question brings up some interesting points.

A problem arises when you also want to use marks for something other than change bars: chapter numbers, for example. In that case you can't just have \beginbar include, say, \mark{\startabar} because you would lose the chapter number information that was also being kept in mark text.

I haven't implemented a general solution to this problem, but I think it could go as follows. Define a \newmark macro that would be invoked for each distinct mark function. In this case \newmark\barmark and \newmark\chaptmark. Then provide a \setmarks macro that defs each of the allocated marks; e.g., in this case \setmarks would be (automatically) defined as follows:

```
\def\setmarks{\mark{%
   \def\noexpand\thebarmark{\barmark}%
   \def\noexpand
            \thechaptmark{\chaptmark}}}
```

then in the text, the usage is

```
\def\barmark{\startabar}\setmarks
```

and

```
\def\chaptmark{...}\setmarks.
```

In the output routine the appropriate marks are first defined and then used:

```
\botmark ...
```

followed by

```
\thebarmark ... and \thechaptmark ...
```

The idea is that the actual mark contains only \defs, which are defined when \botmark (or \topmark, etc.) is referenced.

The second point concerns \specials in general. It does not seem to be universally understood that the random paging mechanism in the dvi file format implicitly proscribes global specials [cf. TUGboat 6, #2 pp. 66–69]. Any global formatting function that uses specials (changing the paper orientation, for example) must repeat the appropriate special command on every output page.

In addition, dvi file printers should be careful not to remember \special parameters between pages.

Letters

Bugs in METAFONTware

To the Editor:

I have discovered a couple of bugs in the METAFONT utility programs having to do with packed files and would like to share this discovery.

The first bug is severe, and makes it virtually impossible to use packed files. It occurs in the Kellerman and Smith implementation (VAX) of PKtoPX (version 2.2), the program which converts packed files to the PXL format most commonly used by device drivers. The bug is in the change file rather than the WEB file, so none of the other implementations are affected. I don't know whether this bug has been previously discovered or not; the number of sites using PK files is still limited. Also, if a driver reads PK files directly, it does not use PKtoPX and the bug does not apply.

The nature of the bug is that, in the Font Directory at the end of the PXL file, the pointers to the glyphs are incorrectly expressed, making it impossible for the driver to find the rasters for the glyph in the main part of the file. According to section 9 of PKtoPX, "The third word of a glyph's directory information contains the number of the word in this PXL file where the raster description for this particular glyph begins, measured from the first word which is numbered zero." Word, in this context, refers to a 32-bit number ("longword" in VAX terminology). The problem is that the changes for the VAX implementation accidentally change this offset from an offset of longwords to an offset of bytes, making the offsets four times greater than they should be. The procedure in question, *pixel_integer*, writes a 32-bit integer to the PXL file and increments a variable called *pxl_loc*, which contains the current offset into the PXL file in longwords. Kellerman and Smith changed this procedure to write the integer as four separate bytes, which is all well and good, but logically the PXL file is still a stream of longwords, so the increment of *pxl_loc* should have been left alone.

To fix the bug, find the following line in PKTOPX.CH:

```
pxl_loc:=pxl_loc+4;
```

and change it to:

```
incr(pxl_loc) ;
```

This will force the variable *pxl_loc* to once again be a longword-count instead of a byte-count.

The other bug, which occurs in GFtoPK (version 1.2), is more subtle, and possibly inconsequential, depending on the driver which is using the packed file. Unfortunately, the error is in the main WEB file, causing it to crop up in *all* implementations.

The problem occurs in module 64, "⟨Scan for bounding box⟩". This code is supposed to take the pixel image of a glyph as generated by METAFONT and strip off rows and columns consisting of all-white pixels from the sides, top, and bottom, resulting in a tighter character. It seems to work in all cases except when there are one or more all-white columns on the left. When this happens, the algorithm fails and the *max_m* variable (and therefore *width*) is set to a value one greater than it should be. PKtoPX cheerfully passes this incorrect value to the font directory in the PXL file. Whether this is a problem or not depends on the individual device driver. The actual raster image is correct — it's just the width that is wrong. If the driver uses the width in the TFM file rather than the one in the PXL file, then the error has absolutely no effect. If the width in the PXL file is used, the character appears one pixel wider than it actually is. In any case, not many characters are affected. I am not familiar enough with METAFONT to describe the conditions which cause all-zero columns to be added on the left. It appears on various characters in various fonts at various magnifications. For an example, do a GFtype of cmr10.300GF and look at the character with a decimal code of 20 (ˇ). After suppression opf the left column, which is all zeros, the actual width of the character should be 10, but a PKtype after GFtoPK is run will show a width of 11. This is the only character in this particular font file which has this problem; other font files have more than one erroneous character, while still others have none at all.

To fix this bug, add the following section to GFTOPK.CH:

```
@x
  min_m := min_m + extra ;
  max_m := min_m + max_m - 1 ;
  height := max_n - min_n + 1 ;
  width := max_m - min_m + 1 ;
@y
  min_m := min_m + extra ;
  max_m := min_m + max_m - 1 - extra ;
  height := max_n - min_n + 1 ;
  width := max_m - min_m + 1 ;
@z
```

Where this section goes in the change file depends on what changes are already in the file, of course. For the Kellerman and Smith VAX version, it goes immediately ahead of:

```
@x
@d preamble_comment == 'GFtoPK 1.2 output'
@d comm_length = 17
@y
@d preamble_comment ==
            'VAX/VMS GFtoPK 1.2.0 output'
@d comm_length = 27
@z
```

E. W. Sewell
Software Engineering Specialist
3822 Hillsdale Lane
Garland, TX 75042-5334
(214) 272-0515

Editor's note: These problems have been communicated to David Kellerman at K&S and to Tom Rokicki at Stanford. David agrees that use of PK files is still very limited, and therefore these programs haven't been given a great amount of exercise. He was already aware of both bugs and said that all corrections would be on the K&S distribution tape no later than October 1, 1986.

Tom Rokicki responded with additional updates to the PK file software; see his note on page 140.

Calendar

1986

Oct 31	Deadline for submission of titles/short abstracts, 1987 TUG Meeting: TEX for the Humanities
Nov 7	Japanese TEX Users Group meeting, Keio University, Keio, Japan (see page 192)
Nov 30	First selection of papers for special program, 1987 TUG Meeting: TEX for the Humanities
Dec 86	METAFONT, TEX and the Humanities, Paris, France (see announcement, page 191)

1987

Jan 19	TUGboat Volume 8, No. 1: Deadline for submission of manuscripts
Jan 19–23	INRIA Course/Structures for Documents, Aussois, Savoie, France (see announcement, page 191)
Jan 31	Deadline for submission of long abstracts, 1987 TUG Meeting: TEX for the Humanities
Jan 31	Deadline for items to be included in preliminary program for general-interest sessions, 1987 TUG Meeting
Feb 28	Final selection of papers for special program, 1987 TUG Meeting: TEX for the Humanities
Apr 15	Deadline for TEX copy for preprints, 1987 TUG Meeting: TEX for the Humanities
May 4	TUGboat Volume 8, No. 2: Deadline for submission of manuscripts (tentative)
Jul 27–31	SIGGRAPH, Anaheim, CA. For information, contact the ACM Conference Office, Chicago, IL; 312-644-6610

1987 TUG Annual Meeting
West Coast, early to mid-August

Day 1	General program; introduction for new members
Days 2–3	Technical program of refereed papers: TEX for the Humanities
Days 2–3	General-interest technical program
Day 3	Deadline for machine-readable copy of refereed papers for Proceedings
Days 4–5	Short course

* * * * *

Sept 14	TUGboat Volume 8, No. 3: Deadline for submission of manuscripts (tentative)
Sept 28–Oct 1	Conference on Electronic/Desktop Publishing, San Francisco, CA. For information, contact National Computer Graphics Association, Fairfax, VA; 703-698-9600

For additional information on the special program, TEX for the Humanities, being offered at the 1987 annual meeting, see page 131. For additional information on other events listed above, contact the TUG office (401-272-9500, ext. 232) unless otherwise noted.

INRIA Course
Structures for Documents

Course directors:

Jacques André, INRIA/IRISA, Rennes
Vincent Quint, INRIA/IMAG, Grenoble

Lecturers:

J. André	INRIA/IRISA, France
R. Furuta	University of Maryland, U.S.A.
R. Ingold	Federal Institute of Technology at Lausanne, Swizerland
V. Joloboff	Bull Corporate Research Center, France
I. Levy	Italsoft, France
P. Norrish	University of Reading, U.K.
V. Quint	INRIA/IMAG, France
R. Southall	XEROX PARC, U.S.A.
J. Virbel	LSI-University of Toulouse, France

Aims

A document may naturally be described as a structure. As an example, a book is split up into chapters, a chapter into sections, sub-sections, paragraphs, and so on. However, this structure might not be so obvious at a deeper level.

This course will focus on

- structured elements of documents, from the computer specialist point of view as well as for the publisher,
- systems and standards for editing, formatting and transmitting structured documents.

The course will be based on lectures of 6 hours dwelling on the computer angle whereas the user point of view will be particularly developed during shorter conferences.

This course concerns researchers and computer scientists involved in the concepts and tools for manipulating structured documents.

Program

Lectures

R. Furuta:	Concepts and models for structured documents (E)
V. Quint:	Systems for manipulating structured documents (F)
V. Joloboff:	Standards and representation of structured documents (F)
B. Reid:	Electronic mail transmission of structured documents (E)

Conferences

J. André:	Why structured documents ? (F)
R. Ingold:	Structure recognition and optical reading (F)
I. Levy:	Logical markup of text for publishers (F)
P. Norrish:	Semantic structures and typology of documents (E)
R. Southall:	Interfaces between the designer and the document (E)
J. Virbel:	Textual structure – a linguistic approach (F)

(E) in English.
(F) in French.

General Information

Location: Aussois, Savoie, France.
Date: January 19–23, 1987.
Registration fees: lodging inclusive
FF 5000
FF 3200 (University)
Registration procedures:
The course attendance is limited to 60 persons.
For further information and registration write to:
INRIA Public Relations Department
Courses and seminars
Domaine de Voluceau, Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France.

METAFONT, TEX and the Humanities

Although initially designed for mathematical and text processing, TEX is suitable for humanities as well, thanks to its ability to process footnotes, exotic languages, fonts, etc.

A seminar on TEX and the Humanities will be organized by GUT (the French *Groupe des Utilisateurs de TEX*) at the end of 1986 or at the beginning of January 87, in Paris, France.

Papers on experiences in using TEX for humanities are welcome. For more details, please contact

Jacques André
IRISA/INRIA
Campus de Beaulieu
F-35042 Rennes, France

Japanese TEX Users Group

Yoshio Ohno
Keio University

The Japanese TEX Users Group was organized at the end of the last year as a special interest group of JSSST (Japan Society for Software Science and Technology). The members of the executive committee are

Yoichi Kawabata (Canon, Inc.),
Nobuko Kishi (IBM Japan, Ltd.),
Ichiro Ogata (Electrotechnical Laboratory),
Yoshio Ohno (Keio University — Secretary),
Toshiharu Ono (ASCII Corporation), and
Nobuo Saito (Keio University — Chairman).

The number of members is approaching 300. Regular meetings have been held twice so far. The talks given are listed below.

First meeting (20 March, at IBM Japan, Ltd.)

- The state of the art of TEX system (Nobuo Saito)
- How to install TEX (Yoshio Ohno)
- TEX on UNIX and MS-DOS (Toshiharu Ono)

Second meeting (17 July, at Keio University)

- Multi-lingual word-processing research at UCLA (Daniel M. Berry)
- Introduction to LATEX (Nhut Nguyen)
- Printing Japanese language using TEX (Masaaki Nagashima and Yoichi Kawabata)
- POSTSCRIPT tutorial (Ichiro Ogata)

About 130 members attended each meeting. The next meeting is scheduled as follows.

Third meeting (7 Nov., 2:30 pm, at Keio University)

- PLAIN macros: how and why do they work? (Ryoichi Kurasawa)
- On the METAFONT system (Yasuki Saito)

For more information, write to

Yoshio Ohno
Institute of Information Science
Keio University
1-1 Hiyoshi 4-chome
Kohoku-ku Yokohama
223 Japan.

TEX in Europe Summer 1986

Barbara Beeton

Two conferences held this summer in Europe were of interest to TEX users: TEX for Scientific Documentation (Strasbourg, June 19–21) and Markup '86 (Luxembourg, July 8–10). I was fortunate in being able to attend both, and to meet many active TEX users and find out about their interests and how they differ from what's happening in North America.

Jacques Désarménien was a knowledgeable and gracious host in Strasbourg. The program was well-organized, and deservedly well-attended.

Most presentations dealt not with the details of TEX, but with systems built around TEX. Particularly in evidence were pre-processors directed at making TEX easier for the novice or casual user.

User environments described by the speakers were principally workstation-based, with preview capability as an integral part of the design. At Berkeley, work is being done to make the preview itself interactive, feeding back corrections into the underlying TEX source file; so far only text is handled in this manner, while mathematical expressions are being treated as single objects.

Several presentations dealt with complete document distribution and delivery systems, both electronic and print-oriented. The form in which a document is stored in such a system could be either TEX source, .DVI files, or some other form, e.g. SGML (see below).

SGML and other non-TEX models were adopted by some researchers because they implement underlying document structures more clearly than does plain TEX. A dissenting view came from Michael Ferguson, who suggested that one should remember that *people* write documents, and he felt that SGML tends to forget this.

Multilingual requirements, both scholarly and mundane, were the basis for several presentations. This general topic, and in particular the problem of using TEX to compose non-English, especially non-Latin alphabet, texts, became the focus of a "birds-of-a-feather" session; see the invitation from Reinhard Wonneberger, page 132, to form a philology special interest group.

Finally, Richard Southall described his experience using METAFONT as a design tool in trying to create a new typeface; his illustrations clearly showed that the assumptions brought by professional type designers to the design process are

not the same as those current in the TEX world. Whether there will emerge skilled type craftsmen who are also fluent in the uses and capabilities of the TEX tools, only time will tell.

Several vendors displayed their products or services, among them Addison-Wesley (MicroTEX and publications), Personal TEX (PC TEX), Te.Co.Graf. (EasyTEX, a pre-TEX, interactive formula processor for IBM PCs), and H. Stürtz (phototypesetter services).

The proceedings of the Strasbourg conference will be published by Springer Verlag this fall, as volume 236 of their series *Lecture Notes in Computer Science*. (A listing of the contents appears below.) The TUG office will have a supply for sale.

This was the second "EuroTEX" conference on the subject of scientific documentation, the first having taken place in Como, Italy, in April 1985. Two future meetings are in the tentative planning stages: Germany in 1987 (to be organized by Bernd Schulze), and England in 1988 (Malcolm Clark).

Markup '86 was sponsored by the Graphic Communications Association and dealt with electronic manuscript preparation, the Standard Generalized Markup Language (SGML) and technical documentation issues. SGML is not a typesetting language, but a methodology for logical document markup that will enable document content to be fully specified for later composition by "one's favorite formatter" (as described by Roberto Minio, of Springer and Carnegie-Mellon, who further insisted that the formatter "must" be TEX). Some reasons were given why a user should want to use SGML rather than a formatter:

- a document may have to be communicated among users who have different formatters;
- separating the user from the formatter allows him to concentrate on the logical structure and content of the document, rather than on its typeset appearance;
- the logically-coded document can be formatted to different style specs for different uses, as needed.

The last two considerations have been implemented in TEX in macro packages such as LATEX; the problem of the first point remains, that it is not easy to translate TEX input to other formatting languages (and vice versa).

The goal of the committee which developed SGML was to create a standard that would be usable by authors and editors, acceptable worldwide, amenable to validation by field testing, and (relatively) independent of problems of technological incompatibility. (A report by Larry Beck, TUG's committee liaison, appears on page 132.)

About ten products based on SGML principles or of interest to SGML users were on display. There was some overlap with the vendors who were in Strasbourg, and at least three of the products were TEX-based. (A description of one of these, the Tiger-TEX Workstation, appears on page 194.)

An active SGML Users' Group exists; information about the organization can be obtained from

SGML Users' Group
c/o Peter V. Howgate, Secretary
Information Services Division
BPCC Graphics Ltd.
Slack Lane
Derby DE3 3FL, U.K.

Contents: Proceedings of "TEX for Scientific Documentation" Strasbourg, June 19–21, 1986

H. Petersen: A TEX-based document factory in a university environment: Processing model, implementation steps, experiences

V. Quint, I. Vatton, H. Bedor: GRIF: An interactive environment for TEX

J. Röhrich: Abstract markup in TEX

R. Southall: Designing a new typeface with METAFONT

R. Wonneberger: "Verheißung und Versprechen", A third generation approach to theological typesetting

The Tiger-TEX Workstation

Susanne Lachmann
Gesycom GmbH, Aachen

Overview

The application target of the TIGER-TEX Workstation is the composition of easy to use tools for the handling of text, images and graphics in one system and the automatic integration of these different types of information into TEX-formatted documents.

"TIGER" is the short form for "TEX-based Image- and Graphics-integration and Enhanced Representation functions".

This article describes the state of implementation, the available tools and further developments of TIGER.

Basic Concepts

The TIGER-TEX Workstation is the realisation of the idea of a workbench for document preparation. The set of tools is subdivided into toolboxes, each of these for the handling of a different type of information as text, image or graphic.

The tools of each toolbox are grouped dependent on the goal of actual work (i.e. editing-tools, formatting-tools, representations-tools etc.). These tool-groups correspond to different data objects which we name as manuscript, typoscript, printscript I and printscript II.

The manuscript-file contains the logically structured document content, that is the text intermixed with logical markup tags, to describe the logical structure of the document. Logical document markup subdivides the document content into logical objects, for example headlines, chapters, paragraphs, sub-paragraphs, footnotes. The manuscript contains no information about the further layout of the document, only the definition of the logical structure and therefore the layout-specification (i.e. the formatting) is independent from any formatter. Every formatting program can be used after the logical markup tags are mapped onto fitting formatting commands and/or macros.

The typoscript-file contains the document content intermixed with formatting-commands and/or -macros. In this sense the normal TEX source file is a typoscript.

The printscript I is the output device independent formatted state of the document, e.g. the TEX-output DVI-file.

The printscript II contains the formatted state of the document for the output on the chosen output device, e.g. the files that are produced by the different TEX-drivers (as VIEW, DVILaser, etc.).

Actual State of Implementation

The TIGER-TEX Workstation at the moment is a well-defined collection of software tools and hardware devices for the processing of each text, image and graphics with the possibility to use additional soft- and hardware within it.

Hardware

The TIGER-TEX Workstation is based on IBM PC XT/AT hardware with Hercules graphics card.

Printing is done on the electronic printing system ELSA (from OLYMPIA). ELSA printing is based on the xerographic process (known from modern copiers) in connection with the use of so-called magneto-optical light switching arrays (instead of the laser beam used by commonly known laser printers). The printing resolution is 300 dots per inch and the printing speed 20 pages DIN A4 (or letter size) per minute.

Other hardware devices can additionally be used within TIGER, for example a flat bed scanner or camera scanner to input pixel information (images) into the system or a OCR-Reader for text input.

Software

The image processing toolbox of TIGER contains a pixel-editor for the creation and pixel-wise editing of image-files, functions for the 1:1 representation on the screen and input routines to get pixel-information from above mentioned external hardware devices.

The TIGER graphics toolbox contains different CAD systems for the creation, editing and representation of vector graphics (e.g. UniCAD, AutoCAD, generic CAD).

Manuscript-preparation of texts in the above described sense of logical structuring is not yet supported by TIGER's text toolbox. For the input and editing of TEX-sources we use an editor which is based on Borland's MS-editor including all MS-editor-functions as word wrap, block management and search and replace functions. The editor produces standard ASCII output (very fine for the TEX-application) and can be used with Borland's Lightning spell checker. The advantage of having the editor in Turbo Pascal source also makes it possible, to create and insert any editor function you would like to have (if you're able to work with Turbo Pascal). This is very important for our work on a structure-oriented Editor for the manuscript-preparation (see further developments).

The text toolbox also contains the PCTEX-formatter, the VIEW-program and e.g. functions for printing the TEX-source and -LOG file.

The physical integration of images and graphics into the text and the output on the electronic printing system ELSA is provided by the in-house-written driver DVIELSA. The driver gets the graphic- and image-files from their toolboxes and integrates them automatically into the text by using the special and midinsert commands of TEX.

Further Developments

What we plan to do and what we are actually working on is an SGML-based author's subsystem, an interactive TEX-environment and the realization of TEX-output on typesetters.

SGML-based Author's Subsystem

In the announced release of TIGER-TEX Workstation the described text-editor will be upgraded by an SGML-Processor and interactive functions to control and lead the author's input of document content considering a specified type of document to be created. SGML is a draft international standard of the ISO (international standardisation organization) for the definition of logical document type structures and the description of logical structures of special documents.

Document types are for examples reports, letters, manuals or articles. The author's subsystem will have a library of available SGML-defined document types and the author will be able to add new document type definitions in an easy way. The process of creating and/or editing document-type-definitions will be controlled by an interactive user-interface, so the author of documents doesn't has to deal with SGML itself.

According to one of these document types, the manuscript editor will lead the author along the predefined logical document type structure and will controll the input.

Also the author will not have to be familiar with TEX if he doesn't want to be. The mapping of his logically structured manuscript onto a TEX-processable typoscript will be automatically done by the system using predefined layout directives, which describe the way to replace the manuscript's logical markup tags by TEX-macros, and this way create the typoscript. The author will have the possibility, to insert self-defined macros as layout directives, but he doesn't have to because there always will be predefined layout directives for every logical object of the document type he is working on.

One of the advantages of such an SGML-based Subsystem is the easy to perform interchanging of manuscripts to other SGML-based systems, where these manuscripts can be mapped onto different macros for other formatting programs without the need of making changes inside the manuscript itself. Another advantage is the possibility, to make TEX useable for people, who are not interested in learning all TEX-features, but are interested in getting fine TEX-formatted output.

Interactive TEX-Environment

We are actually working on a TEX-implementation in an interactive environment on a APOLLO DN300 workstation, based on the GMD-Implentation on an ICL-PERQ-System as described by W. Appelt of the GMD in a further TUGboat-release.

TEX-Output on Typesetters

In order to achieve a professional application of TEX, we hope to finish our work on a TEX-driver for Linotype's LTC 300/500 within this year. Great effort still has to be made to generate the TEX TFM-files (and also the corresponding VIEW/PREVIEW pixel-files) to have the wide spectrum of Linotype's fonts available for TEX.

Institutional Members

Aarhus Universitet, Det Regionale EDB-Center (RECAU), *Aarhus, Denmark*

Addison-Wesley Publishing Company, *Reading, Massachusetts*

American Mathematical Society, *Providence, Rhode Island*

ASCII Corporation, *Tokyo, Japan*

Baumeister College, *Witten, West Germany*

Bell Northern Research, Inc., *Mountain View, California*

California Institute of Technology, *Pasadena, California*

CALMA, *Sunnyvale, California*

Calvin College, *Grand Rapids, Michigan*

Canon, Inc., Office System Center, *Tokyo, Japan*

Carleton University, *Ottawa, Ontario, Canada*

CDS/WordWorks, *Davenport, Iowa*

Centre Inter-Régional de Calcul Électronique, CNRS, *Orsay, France*

Centro Internacional De Mejoramiento De Maiz Y Trigo (CIMMYT), *México, D.F., Mexico*

City University of New York, *New York, New York*

College of St. Thomas, *St. Paul, Minnesota*

Columbia University, Center for Computing Activities, *New York, New York*

COS Information, *Montreal, P. Q., Canada*

Data General Corporation, *Westboro, Massachusetts*

Digital Equipment Corporation, *Nashua, New Hampshire*

Dowell Schlumberger Inc., *Tulsa, Oklahoma*

Edinboro University of Pennsylvania, *Edinboro, Pennsylvania*

Educational Testing Service, *Princeton, New Jersey*

Electricité de France, *Clamart, France*

Environmental Research Institute of Michigan, *Ann Arbor, Michigan*

European Southern Observatory, *Garching bei München, West Germany*

Ford Aerospace & Communications Corporation, *Palo Alto, California*

FTL Systems Incorporated, *Toronto, Ontario, Canada*

General Motors Research Laboratories, *Warren, Michigan*

Geophysical Company of Norway A/S, *Stavanger, Norway*

Grumman Corporation, *Bethpage, New York*

GTE Laboratories, *Waltham, Massachusetts*

Hartford Graduate Center, *Hartford, Connecticut*

Harvard University, *Cambridge, Massachusetts*

Hewlett-Packard Co., *Boise, Idaho*

Humboldt State University, *Arcata, California*

IBM Corporation, Scientific Center, *Palo Alto, California*

Illinois Bell, *Chicago, Illinois*

Illinois Institute of Technology, *Chicago, Illinois*

Imagen, *Santa Clara, California*

Institute for Advanced Study, *Princeton, New Jersey*

Institute for Defense Analyses, Communications Research Division, *Princeton, New Jersey*

Intergraph Corporation, *Huntsville, Alabama*

Intevep S. A., *Caracas, Venezuela*

Istituto di Cibernetica, Università degli Studi, *Milan, Italy*

Kuwait Institute for Scientific Research, *Safat, Kuwait*

Los Alamos National Laboratory, University of California, *Los Alamos, New Mexico*

Marquette University, *Milwaukee, Wisconsin*

Massachusetts Institute of Technology, Artificial Intelligence Laboratory, *Cambridge, Massachusetts*

Mathematical Reviews, American Mathematical Society, *Ann Arbor, Michigan*

McGill University, *Montreal, Quebec, Canada*

McGraw-Hill, Inc., *Englewood, Colorado*

National Center for Atmospheric Research, *Boulder, Colorado*

National Institutes of Health, *Bethesda, Maryland*

National Research Council Canada, *Ottawa, Ontario, Canada*

Online Computer Library Center, Inc. (OCLC), *Dublin, Ohio*

Northeastern University, *Academic Computing Services, Boston, Massachusetts*

Personal TEX, Incorporated, *Mill Valley, California*

QMS, Inc, *Mobile, Alabama*

Queens College, *Flushing, New York*

RE/SPEC, Inc., *Rapid City, South Dakota*

Ruhr Universität Bochum, *Bochum, West Germany*

Rutgers University, Hill Center, *Piscataway, New Jersey*

St. Albans School, *Mount St. Alban, Washington, D.C.*

Sandia National Laboratories, *Albuquerque, New Mexico*

SAS Institute, *Cary, North Carolina*

Schlumberger Offshore Services, *New Orleans, Louisiana*

Schlumberger Well Services, *Houston, Texas*

Science Applications International Corp., *Oak Ridge, Tennessee*

I. P. Sharp Associates, *Palo Alto, California*

Smithsonian Astrophysical Observatory, Computation Facility, *Cambridge, Massachusetts*

Software Research Associates, *Tokyo, Japan*

Sony Corporation, *Atsugi, Japan*

Springer-Verlag, *Heidelberg, West Germany*

Stanford Linear Accelerator Center (SLAC), *Stanford, California*

Stanford University, Computer Science Department, *Stanford, California*

Stanford University, ITS Graphics & Computer Systems, *Stanford, California*

State University of New York, *Stony Brook, New York*

Syracuse University, *Syracuse, New York*

Talaris Systems, Inc., *San Diego, California*

Texas A & M University, Department of Computer Science, *College Station, Texas*

Texas A & M University, Department of Mathematics, *College Station, Texas*

Texas Accelerator Center, *The Woodlands, Texas*

Textset, Inc., *Ann Arbor, Michigan*

TRW, Inc., *Redondo Beach, California*

Tufts University, *Medford, Massachusetts*

TYX Corporation, *Reston, Virginia*

University of British Columbia, *Vancouver, British Columbia, Canada*

University of Calgary, *Calgary, Alberta, Canada*

University of California, Berkeley, Computer Science Division, *Berkeley, California*

University of California, Berkeley, Computing Services, *Berkeley, California*

University of California, San Francisco, *San Francisco, California*

University of Chicago, Computation Center, *Chicago, Illinois*

University of Chicago, Computer Science Department, *Chicago, Illinois*

University of Chicago, Graduate School of Business, *Chicago, Illinois*

University of Delaware, *Newark, Delaware*

University of Glasgow, *Glasgow, Scotland*

University of Groningen, *Groningen, The Netherlands*

University of Illinois at Chicago, *Computer Center, Chicago, Illinois*

University of Kansas, Academic Computing Services, *Lawrence, Kansas*

University of Maryland, *College Park, Maryland*

University of Massachusetts, *Amherst, Massachusetts*

University of North Carolina, School of Public Health, *Chapel Hill, North Carolina*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

University of Ottawa, *Ottawa, Ontario, Canada*

University of Southern California, Information Sciences Institute, *Marina del Rey, California*

University of Tennessee at Knoxville, *Department of Electrical Engineering, Knoxville, Tennessee*

University of Texas at Austin, Physics Department, *Austin, Texas*

University of Texas at Dallas, Center for Space Science, *Dallas, Texas*

University of Washington, Department of Computer Science, *Seattle, Washington*

University of Western Australia, Regional Computing Centre, *Nedlands, Australia*

University of Wisconsin, Academic Computing Center, *Madison, Wisconsin*

University of York, *Heslington, York, England*

Vanderbilt University, *Nashville, Tennessee*

Washington State University, *Pullman, Washington*

Index of Advertisers

Request for Information

The TEX Users Group maintains a database and publishes a membership list containing information about the equipment on which members' organizations plan to or have installed TEX, and about the applications for which TEX would be used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TEX and the hardware on which it runs or is being installed. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, you may indicate that member's name, and the information will be repeated.

If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX Users Group
  P. O. Box 594
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers* direct payment to the TEX Users Group, account #002-031375, at:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903-2449, U.S.A.
- *General correspondence* about TUG should be addressed to:
  TEX Users Group
  P. O. Box 9506
  Providence, Rhode Island 02940-9506, U.S.A.

Name: _____
Home [ ]
Bus. [ ] Address: _____
_____
_____
_____

QTY	ITEM	AMOUNT
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **North America** New (first-time): [ ] $30.00 each Renewal: [ ] $40.00; [ ] $30.00 – reduced rate if renewed before January 31, 1987	
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **Outside North America** New (first-time): [ ] $40.00 each Renewal: [ ] $45.00; [ ] $40.00 – reduced rate if renewed before January 31, 1987	
	TUGboat back issues, $15.00 per issue, circle issue(s) desired: 1980 (v. 1) #1 1981 (v. 2) #1, 2, 3 1982 (v. 3) #1, 2 1983 (v. 4) #1, 2 1984 (v. 5) #1, 2 1985 (v. 6) #1, 2, 3 1986 (v. 7) #1, 2, 3	

Air mail postage is included in the rates for all subscriptions and memberships outside North America.
Quantity discounts available on request.

TOTAL ENCLOSED: _____
(*Prepayment in U.S. dollars required*)

* * * *

Membership List Information

Institution (if not part of address):

Title:
Phone:
Network address: [ ] Arpanet [ ] BITnet
[ ] CSnet [ ] uucp

Specific applications or reason for interest in TEX:

My installation can offer the following software or technical support to TUG:

Please list high-level TEX users at your site who would not mind being contacted for information; give name, address, and telephone.

Date:
Status of TEX: [ ] Under consideration
[ ] Being installed
[ ] Up and running since
Approximate number of users:
Version of TEX: [ ] SAIL
Pascal: [ ] TEX82 [ ] TEX80
[ ] Other (describe)

From whom obtained:

Hardware on which TEX is to be used:

Computer(s)	Operating system(s)	Output device(s)

Please answer the following questions regarding output devices used with TEX if this form has never been filled out for your site, or if you have new information. Use a separate form for each output device.

Name _____ Institution _____

A. Output device information
Device name
Model
1. Knowledgeable contact at your site
Name
Telephone
2. Device resolution (dots/inch)
3. Print speed (average feet/minute in graphics mode)
4. Physical size of device (height, width, depth)
5. Purchase price
6. Device type
[ ] photographic [ ] electrostatic
[ ] impact [ ] other (describe)
7. Paper feed [ ] tractor feed
[ ] friction, continuous form
[ ] friction, sheet feed [ ] other (describe)
8. Paper characteristics
a. Paper type required by device
[ ] plain [ ] electrostatic
[ ] photographic [ ] other (describe)
b. Special forms that can be used [ ] none
[ ] preprinted one-part [ ] multi-part
[ ] card stock [ ] other (describe)
c. Paper dimensions (width, length)
maximum
usable
9. Print mode
[ ] Character: ( ) Ascii ( ) Other
[ ] Graphics [ ] Both char/graphics
10. Reliability of device
[ ] Good [ ] Fair [ ] Poor
11. Maintenance required
[ ] Heavy [ ] Medium [ ] Light
12. Recommended usage level
[ ] Heavy [ ] Medium [ ] Light
13. Manufacturer information
a. Manufacturer name
Contact person
Address
Telephone
b. Delivery time
c. Service [ ] Reliable [ ] Unreliable

B. Computer to which this device is interfaced
1. Computer name
2. Model
3. Type of architecture *
4. Operating system

C. Output device driver software
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)
1. Separate interface hardware not needed because:
[ ] Output device is run off-line
[ ] O/D contains user-programmable micro
[ ] Decided to drive O/D direct from host
2. Name of interface device (if more than one, specify for each)
3. Manufacturer information
a. Manufacturer name
Contact person
Address
Telephone
b. Delivery time
c. Purchase price
4. Modifications
[ ] Specified by Stanford
[ ] Designed/built in-house
[ ] Other (explain)
5. Software for interface device
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

E. Fonts being used
[ ] Computer Modern
[ ] Fonts supplied by manufacturer
[ ] Other (explain)
1. From whom were fonts obtained?
2. Are you using Metafont? [ ] Yes [ ] No

F. What are the strong points of your output device?

G. What are its drawbacks and how have you dealt with them?

H. Comments – overview of output device

* If your computer is "software compatible" with another type (e.g. Amdahl with IBM 370), indicate the type here.

TEX Order Form

The current versions of the public domain TEX software, as produced by Stanford University, are available from Maria Code by special arrangement with the Computer Science Department.

Several versions of the distribution tape are available. The generic ASCII and EBCDIC tapes will require a Pascal compiler at your installation. Each tape contains the source of TEX, and WEB (a precompiler language in which TEX is written), and METAFONT. It also contains font descriptions for Computer Modern, macros for AMS-TEX, LATEX, SliTEX and HP TEX, some sample "change files", and many other odds and ends.

Ready-to-run versions of TEX are available for DEC VAX/VMS, IBM VM/CMS and DEC 20/TOPS-20 formats. They contain everything on the generic tape as well as the compiled programs. This means that you will not need a Pascal compiler unless you want to make source changes. Order these tapes if and only if you have one of these systems.

The font tapes contain GF files for the Computer Modern fonts. While it is possible to generate these files yourself, it will save you a lot of CPU time to get them on the tape.

The price of the tapes now includes the cost of the tape reels. Either 1200′ or 2400′ reels will be used depending on the needed capacity. If you order a distribution tape and a font tape, they will most likely be put on a single 2400′ foot reel. All tapes are 1600 bpi.

Please take care to fill in the order form carefully. Note that postage (other than domestic book rate, which is free) is based on the total weight and postal class which you select. Sales tax is added for orders with a shipping address in California.

The order form contains a place to record the name and telephone number of the person who will actually use TEX. This should *not* be someone in the purchasing department.

Make checks payable to *Maria Code*. Export orders must send checks which are drawn on a US bank. International money orders are fine. Purchase orders are accepted if your company has a policy of prompt payment (30 days maximum).

Your order will be filled with the current versions of software and manuals at the time it is received. Since some versions are "pre-announced", please indicate if you want to wait for a specific version.

Telephone calls are discouraged, but if you must call, please do so between 9:30 a.m. and 2:30 p.m. West Coast time. The number for Maria Code is (408) 735-8006. Do not call for advice or technical assistance since no one is there who can help you. You may try Stanford or some other of the helpful people whose names appear in TUGboat.

9/86

TEX Order Form

Distribution tapes:

ASCII generic format ____
EBCDIC generic format ____
VAX/VMS Backup format ____
DEC 20/TOPS-20 Dumper format ____
IBM VM/CMS format ____

Tape prices: $92 for first tape, $72 for each additional tape

Font tapes (GF files):

200/240 dpi CM fonts ____
300 dpi CM fonts ____

Allow 2 lbs shipping weight for each tape ordered.

Documents:	Price $	Weight	Quantity
TEXbook (vol. A) softcover	25.00	2	____
TEX: The Program (vol. B)	37.00	4	____
METAFONTbook (vol. C) softcover	22.00	2	____
METAFONT the Program (vol. D)	37.00	4	____
Computer Modern Typefaces (vol. E)	37.00	4	____
WEB language *	12.00	1	____
TEXware *	10.00	1	____
BibTEX *	10.00	1	____
Torture Test for TEX *	8.00	1	____
Torture Test for METAFONT *	8.00	1	____
LATEX – document preparation system	25.00	2	____

* published by Stanford University

Payment calculation:

Number of tapes ordered _____ Total price for tapes ____
Number of documents ordered _____ Total price for documents ____
Add the 2 lines above ____
Orders from within California: Add **sales tax** for your location ____

Shipping charges: (for domestic book rate, skip this section)

Total weight of tapes and books _____ lbs.
Check type of shipping and note rate:

____ domestic priority mail: rate $ 1.00/lb.
____ air mail to Canada and Mexico: rate $ 1.50/lb.
____ export surface mail (all countries): rate $ 1.00/lb.
____ air mail to Europe, South America: rate $ 4.00/lb.
____ air mail to Far East, Africa, Israel: rate $ 6.00/lb.

Multiply total weight by shipping rate. Enter **shipping charges:**

Total charges: (add charges for materials, tax and shipping)

Methods of payment: Check drawn on a US bank. Make payable to Maria Code.
International money order.
Purchase order (maximum 30 days allowed for payment)

Send order to: Maria Code, Data Processing Services,
1371 Sydney Drive, Sunnyvale, CA 94087

Name and address for shipment: _____

_____

_____

Contact person (if different from above): _____

Telephone: _____

9/86

TEXTSET, Inc. 416 Fourth Street • P.O. Box 7993 • Ann Arbor • MI • 48107 • (313) 996-3566
technical composition software and services

ANNOUNCEMENT

TEXTSET, Inc. of Ann Arbor, Michigan

has changed its name to

ArborText™ Inc.

as of November 1, 1986.

As **ArborText** we will continue to offer the same professional TEX support and product line on which we have built our reputation. We are also proud to introduce **The Publisher**™, a future product family that will revolutionize text processing on microcomputers and engineering workstations.

Watch for **The Publisher**™ from **ArborText**™ in 1987.

Please correct your records.

COMMITTED.

TEX-Related Publications Available from TUG

For prices and ordering instructions, see separate order form. Current forms may be obtained by writing to TUG, P. O. Box 9506, Providence, RI 02940, U.S.A., or calling (401) 272-9500, ext. 232.

Computers and Typesetting by Donald E. Knuth
A hardcover library of five volumes containing the authoritative documentation on TEX and METAFONT. Volumes A–E are described below.

Published by Addison-Wesley Publishing Co., Inc., Reading, Mass., 1986.

The TEXbook by Donald E. Knuth
From the cover: "TEX represents the state-of-the-art in computer typesetting. It is particularly valuable where the document, article, or book to be produced contains a lot of mathematics, and where the user is concerned about typographic quality. TEX software offers both writers and publishers the opportunity to produce technical text, in an attractive form, with the speed and efficiency of a computer system.

"Novice and expert users alike will gain from *The TEXbook* the level of information they seek. ... The novice need not learn much about TEX to prepare a simple manuscript with it. But for the preparation of more complex documents, *The TEXbook* contains all the detail required."

Volume A of **Computers and Typesetting**. Published jointly by Addison-Wesley Publishing Co., Inc., Reading, Mass., and American Mathematical Society, Providence, R.I., 1986 edition. (Also available in softcover.)

TEX: The Program by Donald E. Knuth
From the introduction: "The main purpose of the... program is to explain the algorithms of TEX as clearly as possible. As a result, the program will not necessarily be very efficient when a particular Pascal compiler has translated it into a particular machine language. However, the program has been written so that it can be tuned to run efficiently in a wide variety of operating environments by making comparatively few changes. Such flexibility is possible because the documentation... is written in the WEB language, which is at a higher level than Pascal; the preprocessing step that converts WEB to Pascal is able to introduce most of the necessary refinements. Semi-automatic translation to other languages is also feasible, because the program... does not make extensive use of features that are peculiar to Pascal."

Volume B of **Computers and Typesetting**. Published by Addison-Wesley Publishing Co., Inc., Reading, Mass., 1986.

The METAFONTbook by Donald E. Knuth
From the preface: "METAFONT is a system for the design of alphabets suited to raster-based devices that print or display text. The characters [used to set the book] were all designed with METAFONT, in a completely precise way; and they were developed rather hastily by the author of the system, who is a rank amateur at such things. It seems clear that further work with METAFONT has the potential of producing typefaces of real beauty. This manual has been written for people who would like to help advance the art of mathematical type design."

Volume C of **Computers and Typesetting**. Published jointly by Addison-Wesley Publishing Co., Inc., Reading, Mass., and American Mathematical Society, Providence, R.I., 1986; available Summer '86. (Also available in softcover.)

METAFONT: The Program by Donald E. Knuth
The complete source code for METAFONT.

Volume D of **Computers and Typesetting**. Published by Addison-Wesley Publishing Co., Inc., Reading, Mass., 1986; available Summer '86.

Computer Modern Typefaces by Donald E. Knuth
This reference work contains the full METAFONT descriptions of the letters and symbols which comprise the Computer Modern family of typefaces. The preface to an earlier report on this subject stated, "It is hoped that some readers will be inspired to make similar definitions of other important families of fonts. The bulk of this [document] consists of... METAFONT programs for the various symbols needed, and as such it is pretty boring, but there are some nice illustrations."

Volume E of **Computers and Typesetting**. Published by Addison-Wesley Publishing Co., Inc., Reading, Mass., 1986; available Summer '86.

TUGBOAT

TUGboat is the newsletter of the TEX Users Group. It contains communications from the Stanford TEX Project; articles of interest to installers and users of TEX and METAFONT; reports of activity at distribution centers and user sites; macros, problems, questions and answers; a calendar of TEX and TUG-related events; and official TUG business.

Three issues are planned for 1987. Memberships and subscriptions are accepted on a calendar-year basis only. All back issues are available.

First Grade TEX: A Beginner's TEX Manual
by Arthur L. Samuel
From the introduction: "This is an introductory ready-reference TEX82 manual for the beginner who would like to do First Grade TEX work. Only the most basic features of the TEX system are discussed in detail. Other features are summarized in an appendix and references are given to the more complete documentation available elsewhere."

Originally published as Stanford Computer Science Department Report No. STAN-CS-83-985. Reprinted with permission and distributed by TUG.

The Joy of TEX by Michael Spivak
This is the user's guide for $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX, an extension of Donald Knuth's typesetting program TEX. $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX and *The Joy of TEX* were written to allow simplified input of mathematical material, and to permit the output to be formatted according to various preset style specifications. Use of $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX requires no expertise in mathematics or in computer science; it requires no more than that TEX itself be available.

Published by the American Mathematical Society, Providence, R.I., 1986.

LATEX: A Document Preparation System
by Leslie Lamport
From the preface: "The LATEX document preparation system is a special version of Donald Knuth's TEX program. ... LATEX adds to TEX a collection of commands that simplify typesetting by letting the user concentrate on the structure of the text rather than on formatting commands. In turning TEX into LATEX, I have tried to convert a highly-tuned racing car into a comfortable family sedan. The family sedan isn't meant to go as fast as a racing car or be as exciting to drive, but it's comfortable and gets you to the grocery store with no fuss. However, the LATEX sedan has all the power of TEX hidden under its hood, and the more adventurous driver can do everything with it that he can with TEX."

Published by Addison-Wesley Publishing Co., Inc., Reading, Mass., 1985.

An Introduction to LATEX by Michael Urban
From the introduction: "This paper is intended to introduce you to the LATEX document preparation system, and to get you working with LATEX as fast as possible. It is *not* a complete reference work on LATEX."

Originally prepared for the TRW Software Productivity Project, 1986. Reprinted with permission and distributed by TUG.

LATEX Command Summary by L. Botway and C. Biemesderfer
Introduction: "This summary ostensibly represents LATEX control sequences for the software implemented at STScI [Space Telescope Science Institute] as of December 6, 1985 (TEX 1.3, LATEX 2.08)."

Originally prepared for internal use at STScI; reprinted with permission and distributed by TUG.

VAX Language-Sensitive Editor (LSEDIT) Quick Reference Guide for use with the LATEX Environment
From the statement of scope and intent: "This guide presents information about the Language-Sensitive Editor (LSEDIT) and the environment defined for LATEX. It is a supplement to the *VAX Language-Sensitive Editor User's Guide* and *LATEX, A Document Preparation System, User's Guide & Reference Manual*. This guide acts as a summary and memory refresher for the commands and functions covered in [these manuals]. It is *not* intended to replace either of [them]."

Originally prepared for internal use at Lear Siegler, Inc., Instrument Division, Grand Rapids, Michigan; reprinted by permission and distributed by TUG.

Proceedings of the First European Conference on TEX for Scientific Documentation, Como, Italy, May 16–17, 1985 Dario Lucarella, Editor
From the preface: "The aim of the Conference was to provide a state-of-the-art survey of current research activities and the latest applications that are growing around TEX. The topics covered ... concern the following fields: Documentation systems based on TEX; TEX as a tool for authors; Customization of TEX for non-English languages; Hyphenation; Standardization problems; Facilities for interactive entering and retrieving of formulae; METAFONT and font design; Implementation of TEX, METAFONT and drivers." All papers are in English.

Published by Addison-Wesley Publishing Co., Inc., Reading, Mass., 1985.

Mathematics into Type by Ellen Swanson
This book covers the publication of mathematics from manuscript to printed book or journal, with emphasis on the preparation of copy for the compositor and the proofreading and makeup of the publication. It will be useful to the author who is directly concerned in the editing of his book, and it should benefit any author who is preparing a manuscript for publication.

Published by the American Mathematical Society, Providence, R.I., 1982.

Available Now!

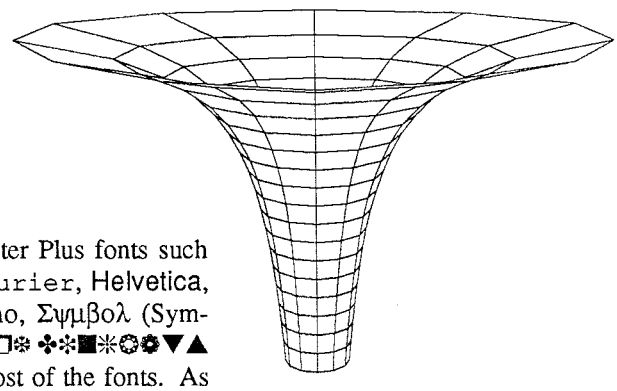
MacTEX™

A complete typesetting system for the Macintosh Plus

MacTEX is the first full version of TEX available for the Macintosh Plus computer. A full implementation, it includes both the PLAIN and LATEX macro packages with LaserWriter and LaserWriter Plus fonts and the standard TEX fonts. With the package, you also get INITEX and all the sources for the macro packages PLAIN and LATEX. With this you can generate your own format files on top of an already existing one, or create a completely new one and select it for loading. And **MacTEX** has passed the "TEX torture test", so you can be sure that the output you get is to the exacting TEX standards.

MacTEX is more than TEX itself. It also contains a built-in multiple file editor and a screen previewer which lets you view the output before printing. Also in the program is the LaserWriter AppleTalk driver to print on a LaserWriter, LaserWriter Plus or any other AppleTalk printer such as a Linotronic 300 (a 2,540 dot per inch laser typesetter). This means that you'll never have to leave the program.

MacTEX will also let you send POSTSCRIPT directly thus allowing effects such as grey text or rotated text. (As you can see above, "Available Now!" is rotated 25 degrees, yet it was set using TEX.) You can also include POSTSCRIPT text files or MacPaint files for graphics. The example on the right was included in printing and the text file containing the POSTSCRIPT code was created using **MacTEX**'s built-in editor.



MacTEX supports all the LaserWriter and LaserWriter Plus fonts such as ITC Avant Garde Gothic, ITC Bookman, Courier, Helvetica, Helvetica-Narrow, New Century Schoolbook, Palatino, Σψμβολ (Symbol), Times-Roman, *Zapf Chancery Medium Italic*, and ✷✪❑✻✣✼■✳✪✪▼▲ (Zapf Dingbats) with **bold**, *italic* and ***bold-italic*** for most of the fonts. As well, five LATEX fonts and three math fonts, *amsy*, *ammi*, and *amex*, are included for full math typesetting and have been converted from pixels into outline form allowing them to be scaled to any size without seeing jagged edges on the letters.

- **Editor**: Standard mouse-driven Macintosh text editor with Cut, Copy, Paste, Clear plus Find, Find Same, Replace, Replace All, and Goto Line.
- **TEX**: Standard TEX with on-screen buttons for error handling.
- **Previewer**: Goto previous page, next page or jump to one at random. View full page (3:1 reduction), full size (1:1) or double size (1:2). Dump the page image to an ImageWriter for proofing.
- **Printer**: Print the *.dvi* file on any LaserWriter on AppleTalk, or download a POSTSCRIPT file. You may also save the *.dvi* file as a POSTSCRIPT text file and download it later.

Also available: **French MacTEX**, containing complete hyphenation tables for the French language.
Font Metric files (*.tfm* files) for other downloadable Adobe Fonts.

Order Now: American Express accepted.

FTL systems Inc.
(416) 487-2142
234 Eglinton Ave. E., Suite 205
Toronto, Ontario
Canada M4P 1K5

This page was set using only **MacTEX** and printed on a LaserWriter Plus.

MACROTEX:

A TEX toolkit. An extensive set of macros that function in a Plain TEX environment. The functionality of a full macro package without restricting your complete access to TEX. Compatible with preexisting macros or macros yet to be designed.

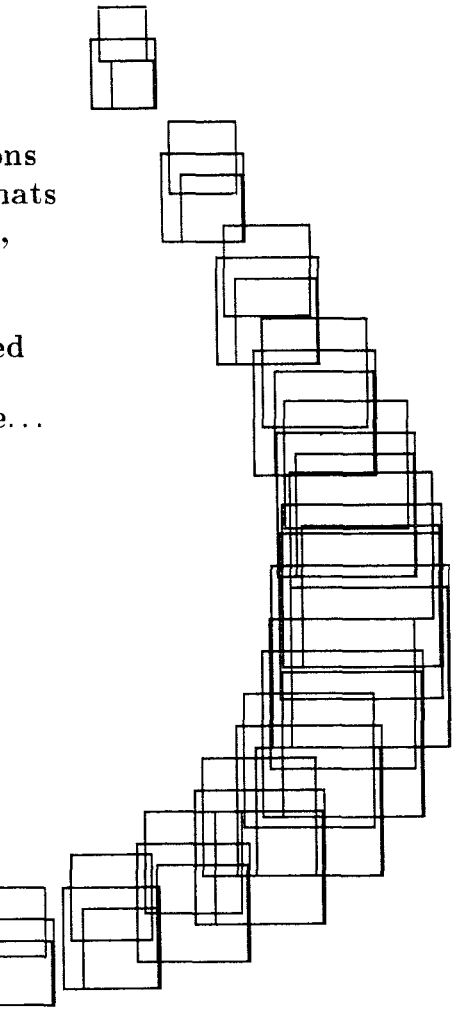


Table of Contents Generation ◇ Index production ◇ Cross Referencing ◇ Symbolic names for Equations ◇ Bibliography production ◇ Paper, Letter Formats ◇ Autonumbering Chapters, Sections, Equations, Figures ◇ Figure macros ◇ Margin notes ◇ Inserts: double column, partial page ◇ Itemization, automatically numbered or lettered ◇ Table Macros ◇ Diagonal lines ◇ Diamond shapes of any size or slant ◇ And many more...

Documentation includes directions for MACROTEX use as well as the complete MACROTEX code annotated with suggestions for customization. Diskette included for PC or Macintosh users.
Price: $50 for individual users, site license available for multiple users. Please write or call for complete descriptive brochure and order form.

TEXnology, Inc.

Amy Hendrickson, Consultant

57 Longwood Avenue, Brookline, MA 02146 *(617) 738-8029*

Custom Macro Writing · Instruction · Book Production

PreTEXt: Don't Print Until You're Ready

The PreTEXt™ software program provides complete page previewing capabilities for TEX™ on the new Talaris 7800 terminal, the **first** page previewing terminal designed for laser printers. PreTEXt allows you to view a TEX document on the terminal exactly as it will appear when it is printed on a Talaris laser printer or any other TEX output device. If the page doesn't look quite right the first time, you can change the TEX input file until it does. Since you print the file when you're sure it's ready, you save time, as well as paper.

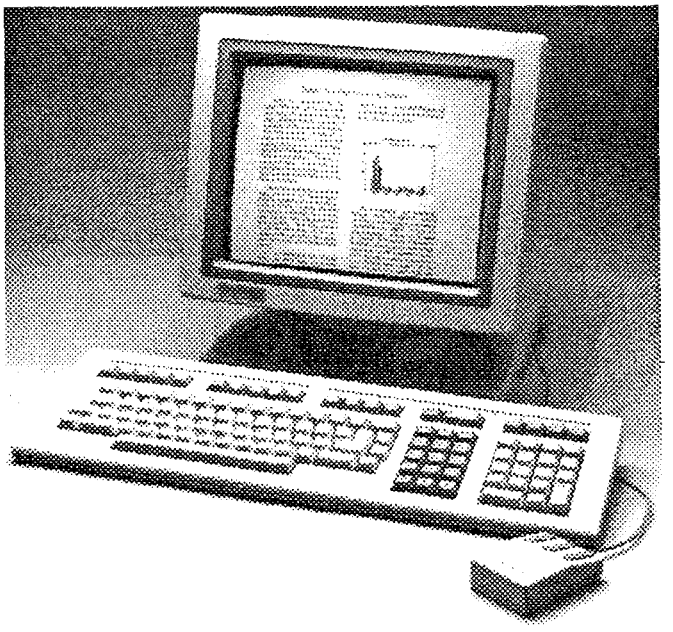
Of course you've heard of other vendors offering DVI to VDU programs. But *only* PreTEXt allows you to view those documents with Talaris's METAFONT-generated library of more than 400 TEX and LATEX Computer Modern fonts at the resolution of the Talaris 7800—so you can see exactly what your printed page will look like. For speedy DVI file processing, the Talaris 7800 also features 28 terminal-resident CM fonts. Fonts not resident in ROM are dynamically down-loaded to the Talaris 7800 from the host computer.

And *only* Talaris lets you view integrated text and Tektronix graphics on the screen. PreTEXt processes input from both TEX DVI files and Tektronix 4014 graphics display lists, which gives you the ability to merge graphics files with your text; graphics from ISSCO®, Precision Visuals®, Megatek™, SAS Institute, and others.

PreTEXt and the Talaris 7800 with Talaris's TEXsupport, QDRIVE®, and a Talaris 800, 1200, or 2400 laser printer provides a complete document typesetting system.

PreTEXt is available for systems running DEC VAX/VMS (Version 4) and Berkeley 4.3 UNIX™. For compatibility with the CM fonts, you must be using TEX Version 2.0.

For further information, call or write Talaris Systems Inc., 5160 Carroll Canyon Road, P.O. Box 261580, San Diego, CA 92126; (619) 587-0787.



The Talaris 7800 page previewing terminal and Talaris's new PreTEXt software program permit you to view TEX documents in the same CM fonts that you get on the printed page.